New Aspects of Beyond Worst-Case Analysis

Colin White

CMU-CS-18-123

October 2018

Department of Computer Science
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Maria-Florina Balcan (Chair)
Anupam Gupta
David Woodruff
Avrim Blum (Toyota Technological Institute at Chicago)
Yury Makarychev (Toyota Technological Institute at Chicago)

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

This research was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate (NDSEG) Program, the National Science Foundation under grant numbers CCF-1451177, CCF-1422910, and IIS-1618714, a John Woodruff Simpson Fellowship, and an Amherst Memorial Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. Government, or any other entity.

Keywords: Beyond Worst-Case Analysis, Clustering, k-means, Perturbation Resilience, Distributed Computation, Algorithm Configuration

Dedicated to my mom.

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 continue the line of work of BWCA for clustering and distributed learning by making contributions in several areas. Specifically, we focus on three main problems and models.

- Clustering is one problem that has benefited greatly from BWCA. The *perturbation resilience* assumption proposed by Bilu and Linial (2011) states that the optimal clustering does not change under small perturbations to the input distances. We design efficient algorithms that output optimal or near-optimal clusterings for the canonical k-center objective (in both symmetric and asymmetric forms) under perturbation resilience. Our results are tight up to the level of perturbation resilience assumed. Next, we give robust algorithms with guarantees even if only part of the data satisfies the perturbation resilience assumption.
- Clustering has many varied applications in practice, and an algorithm may have drastically different performance between two applications. To reconcile this fact, we consider a data-driven approach, 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 linkage algorithms with dynamic programming, as well as local search algorithms with initialization, generalizing the k-means algorithm. We bound the pseudo-dimension of these classes, which leads to computational- and sample-efficient meta-algorithms for some of the algorithm classes we study.
- As datasets become larger and more complex, 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 design general and robust distributed clustering algorithms.

Acknowledgments

First and foremost, I would like to thank my advisor, Nina Balcan. This thesis would not have been possible without Nina's guidance the last four years. Nina's enthusiasm for research and ability to come up with fascinating project ideas is unparalleled. Nina helped me drastically improve my technical writing and presenting skills by giving me many opportunities to hone these skills. She also gave me countless professional opportunities such as invitations to workshops and talks, and introducing me to famous researchers. I thank Nina for the plethora of knowledge she has been nice enough to share with me during my time at CMU.

I give a huge thanks to the other members of my thesis committee, Avrim Blum, Anupam Gupta, David Woodruff, and Yury Makarychev, for interesting research meetings and insightful comments about my work. I thank Yury Makarychev for inviting me to TTIC for the summer of 2017. I thoroughly enjoyed my time at TTIC, working with Yury, Kostya Makarychev, and Haris Angelidakis.

I thank all of my other collaborators whom I have had the pleasure to work with, Nika Haghtalab, Travis Dick, Krishna Pillutla, Mu Li, Alex Smola, Ellen Vitercik, Ainesh Bakshi, Pranjal Awasthi, Vaishnavh Nagarajan, Haris Angelidakis, Kostya Makarychev, and Sunny Gakhar. They have all made research meetings and paper writing fun and exciting. I also want to thank my past and current officemates, Akshay, Sarah, Nic, and Aurick, for helpful discussions, coffee breaks at just the right times, and making my CMU experience more enjoyable overall. More broadly, I am grateful for the students in the theory group, and the computer science department as a whole, for making my time here remarkable. I thank the administrative staff, in particular Deb Cavlovich, Catherine Copetas, and Amy Protos, for being extremely helpful and keeping the department running smoothly.

I want to thank all of my friends at Ascend, the Pharaoh Hounds, the Ascend run club, the J Crew, the supa squad, and all my other Pittsburgh, Amherst, and Evanston friends for their constant support and kindness. I do not know how I would have made it without their help.

Last but not least, I thank my family for their endless encouragement the past four years. Carrie, for going on runs and going to great vegan restaurants with me, Dad, for taking me out for donuts and being a constant support, Andrew, for high quality memes and keeping me up to date with the real world, Rachel, for remembering all birthdays/holidays and generally keeping the family together, and Grandpa, for great holiday cards and phone conversations. And of course, my mom. She was the starting point of my academic career, for fostering my interest in reading, problem solving, and science, and giving me the confidence to aim big. I will never forget her dedication to Carrie, Andrew, and me, and I owe all of my accomplishments to her.

Contents

List of Figures

List of Tables

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. One of the most famous examples 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. Therefore, worst-case analysis is not helpful in determining which linear programming algorithm is better in practice. Indeed, the worst-case instances for the simplex method seem to be delicate constructions, where variables must be set precisely to force the algorithm to run in exponential time. Although these particular instances dominate worst-case analysis, it is very rare that we see such instances come up naturally in a real-life application. Furthermore, the intricate worst-case constructions seem to break under tiny perturbations. Spielman and Teng made this observation explicit by showing that if an arbitrary input undergoes a small randomized perturbation, the expected runtime of the simplex method is linear in the number of variables [Spielman and Teng, 2004]. They referred to this model as "smoothed analysis". This seminal work 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/or more practical models. Two of the main goals of BWCA are *(1)* to use realistic models to design new theoretical algorithms which perform well in practice, and *(2)* 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. BWCA has seen a huge amount of success in many areas in the last few decades.

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 a wide range of applications including bioinformatics, computer vision, text analysis, and countless others. The underlying goal is to partition a given set of points to maximize similarity within a partition and minimize similarity across different partitions. A common approach to clustering is to consider an objective function over all possible partitionings and seek solutions that are optimal according to the objec-

tive. Given a set of points (and a distance metric), common clustering objectives include finding k centers to minimize the sum of the distance from each point to its closest center (k-median), or to minimize the maximum distance from a point to its closest center (k-center).

Traditionally, the theory of clustering has focused on the analysis of worst-case instances. For example, it is well-known the popular objective functions are provably NP-hard to optimize exactly or even approximately (APX-hard) [Gonzalez, 1985, Jain et al., 2002, Lee et al., 2017], so research has focused on finding approximation algorithms. Recently, clustering under beyond worst-case models has become popular, in part because explicitly making real-world assumptions about the input can lead to algorithms which provably return optimal or near-optimal clusterings.

In this thesis, our goal is to continue this strong line of BWCA. We focus on clustering and distributed machine learning problems. We consider three main models of BWCA, and we design new algorithms, prove new structure, and make the assumptions more robust for real-world data. We give a summary of our contributions below.

Clustering under Perturbation Resilience The popular notion of α-perturbation resilience, introduced by Bilu and Linial [2012], 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 symmetric k-median, k-means, and k-center under 2-perturbation resilience [Angelidakis et al., 2017].

In Chapter 2, we show that *any* 2-approximation algorithm for k-center will always return the clusters satisfying 2-perturbation resilience. Since there are well-known 2-approximation algorithms for symmetric k-center, this implies efficient algorithms for clustering under 2-perturbation resilience. Next we design an algorithm which returns the exact solution for asymmetric k-center under 2-perturbation resilience. As asymmetric k-center has a tight $O(\log^*(k))$ approximation factor on worst-case instances, to our knowledge, this is the first problem that is hard to approximate to any constant factor in the worst case, yet can be optimally solved in polynomial time under perturbation resilience for a constant value of α. Then we prove our results are tight by showing symmetric k-center under $(2-\delta)$-perturbation resilience is hard unless $NP = RP$. This is the first tight result quantifying the power of perturbation resilience for a canonical combinatorial optimization problem, i.e., this is the first time the exact value of α for which the problem switched from being NP-hard to efficiently computable has been found.

In this line of work on clustering under perturbation resilience, researchers have developed an array of sophisticated tools exploiting the structural properties of such instances leading to algorithms which can output the optimal solution. However, overly exploiting a BWCA assumption can lead to algorithms that perform poorly when the input data does not exactly satisfy the given assumption. Indeed, recent analyses and technical tools are susceptible to small deviations from the BWCA assumptions which can propagate when just a small fraction of the data does not satisfy the assumption. In Chapter 2, we give a natural continuation of this line of work by providing robust algorithms which give the optimal solution under perturbation resilience, and also perform well when the data is partially perturbation resilient, or not at all perturbation resilient. These

algorithms act as an interpolation between worst-case and beyond worst-case analysis.

The work from Chapter 2 is based on the following papers: Balcan et al. [2016], Balcan and White [2017].

Data-Driven Clustering In Chapter 3, we consider a different approach within BWCA based on the idea that the performance of a given clustering algorithm depends on the specific application at hand, and this may not be known up front. For example, a "typical instance" for clustering protein sequences may look very different from a typical instance for clustering documents in a database, and clustering heuristics perform differently depending on the instance. We use a recently defined PAC-learning framework by Gupta and Roughgarden [2016], in which 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. First we define rich, infinite classes of linkage-based algorithms with a pruning step, and prove tight bounds on the pseudo-dimension of these classes. Then we define a parameterized family of Lloyd's algorithms, with one parameter controlling the initialization step, and another parameter controlling the local search step. We show the sample complexity needed to find parameters close to the optimal parameters over the distribution is low. For both our linkage-based families and our Lloyd's algorithm family, we show computationally efficient algorithms to learn the best algorithm parameters for a specific application.

The work from Chapter 3 is based on the following papers: Balcan et al. [2017, 2018].

Data-Driven Dispatching for Distributed Learning Chapter 4 introduces a BWCA model similar to data-drive clustering, but applied to distributed learning. We study the model of distributed machine learning in which data is dispatched to multiple machines, and there is no communication among machines at training time. The simplest approach is to distribute the data randomly among all machines. Motivated by the fact that similar data points often belong to the same or similar classes, and more generally, classification rules of high accuracy tend to be "locally simple but globally complex" [Vapnik and Bottou, 1993], we propose data-dependent dispatching that takes advantage of such structure by sending similar datapoints to the same machine. For example, a *globally* accurate classification rule may be complicated, but each machine can accurately classify its *local* region with a simple classifier. We provide new dispatching algorithms with provable worst-case guarantees by overcoming novel technical challenges to satisfy important conditions for accurate distributed learning, including fault tolerance and balancedness. This is accomplished by having the dispatcher solve a clustering problem in the input. Finally, we consider the distributed clustering problem. We construct general and robust algorithms for distributed clustering.

The work from Chapter 4 is based on the following papers: Dick et al. [2017], Awasthi et al. [2017].

Tieing it all together Chapters 2, 3, and 4 all go beyond the worst-case in slightly different ways. Recall that worst-case analysis evaluates an algorithm by its performance on just a single instance—the worst one. Chapters 2 and 4 explicitly assume that any instance we see has more structure than a worst-case instance. Then the analysis is still worst-case, but only over the restricted set of structured instances, not the entire input space. This often more accurately portrays the real world; for instance, the restricted set of instances may no longer contain pathological constructions used

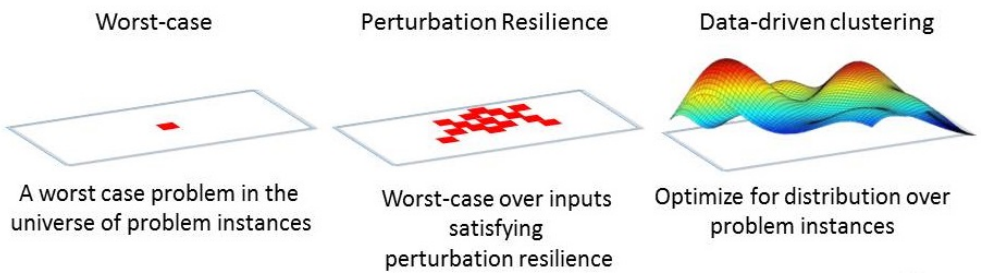


Figure 1.1: Different models in Beyond Worst-Case Analysis

in hardness proofs. Chapter 3 is a slightly different take. In this chapter, we directly optimize for a specific application. The evaluation is now the expected performance over the unknown distribution corresponding to our application, not worst-case. This model is closer to average-case analysis, in which the performance of an algorithm is measured by its average performance over all problem instances. See Figure 1.1 for a summary of the BWCA models we study and their relation to one another.

Chapter 2

k-center Clustering under Perturbation Resilience

2.1 Introduction

Clustering is a fundamental problem in combinatorial optimization with a wide range of applications including bioinformatics, computer vision, text analysis, and countless others. The underlying goal is to partition a given set of points to maximize similarity within a partition and minimize similarity across different partitions. A common approach to clustering is to consider an objective function over all possible partitionings and seek solutions that are optimal according to the objective. Given a set of points (and a distance metric), common clustering objectives include finding k centers to minimize the sum of the distance from each point to its closest center (k-median), or to minimize the maximum distance from a point to its closest center (k-center).

Traditionally, the theory of clustering (and more generally, the theory of algorithms) has focused on the analysis of worst-case instances [Arya et al., 2004, Byrka et al., 2015c, Charikar et al., 1999b, 2001, Chen, 2008, Gonzalez, 1985, Makarychev et al., 2016]. For example, it is well known the popular objective functions are provably NP-hard to optimize exactly or even approximately (APX-hard) [Gonzalez, 1985, Jain et al., 2002, Lee et al., 2017], so research has focused on finding approximation algorithms. While this perspective has led to many elegant approximation algorithms and lower bounds for worst-case instances, it is often overly pessimistic of an algorithm's performance on "typical" instances or real world instances. As explained in Chapter 1, one way to reconcile this fact is to consider the problem of clustering *beyond the worst case*. For example, the popular notion of α-perturbation resilience, introduced by Bilu and Linial [2012], considers instances such that the optimal solution does not change when the input distances are allowed to increase by up to a factor of α. The goals of beyond worst-case analysis (BWCA) are twofold: *(1)* to design new algorithms with strong performance guarantees under the added assumptions [Balcan et al., 2013a, Hardt and Roth, 2013, Kumar et al., 2004, Roughgarden, 2014], and *(2)* to prove strong guarantees under BWCA assumptions for existing algorithms used in practice [Makarychev et al., 2014, Ostrovsky et al., 2012, Spielman and Teng, 2004]. An example of goal *(1)* is a series of work focused on finding exact algorithms for k-median, k-means, and k-center clustering under α-perturbation resilience [Awasthi et al., 2012, Balcan and Liang, 2016, Angelidakis et al., 2017].

The goal in this line of work is to find the minimum value of $\alpha \geq 1$ which admits an efficient algorithm to find the optimal solution for α-perturbation resilient clustering instances. An example of goal *(2)* is the celebrated result by Spielman and Teng [2004] establishing the smoothed linear time complexity of the simplex algorithm.

In approaches for answering goals *(1)* and *(2)*, researchers have developed an array of sophisticated tools exploiting the structural properties of such instances leading to algorithms which can output the optimal solution. However, overly exploiting a BWCA assumption can lead to algorithms that perform poorly when the input data does not exactly satisfy the given assumption. Indeed, recent analyses and technical tools are susceptible to small deviations from the BWCA assumptions which can propagate when just a small fraction of the data does not satisfy the assumption. For example, some recent algorithms make use of a dynamic programming subroutine which crucially need the entire instance to satisfy the specific structure guaranteed by the BWCA assumption. It is increasingly more essential to find algorithms which "gracefully degrade" from the optimal BWCA guarantees down to the worst-case approximation guarantees, as the BWCA assumptions are violated more and more. Another downside of existing approaches is that BWCA assumptions are often not efficiently verifiable. This creates a catch-22 scenario: it is only useful to run the algorithms if the data satisfies certain assumptions, but a user cannot check these assumptions efficiently. For example, by nature of α-perturbation resilience (that the optimal clustering does not change under *all* α-perturbations of the input), it is not known how to test this condition without computing the optimal clustering over $\Omega\left(2^n\right)$ different perturbations. In this chapter, we give a natural continuation of this line of work by providing robust algorithms which give the optimal solution under perturbation resilience, and also perform well when the data is partially perturbation resilient, or not at all perturbation resilient. These algorithms act as an interpolation between worst-case and beyond worst-case analysis.

2.1.1 Results and techniques

In this work, we study symmetric/asymmetric k-center under perturbation resilience. Our algorithms 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. In most cases, our algorithms are natural modifications to existing approximation algorithms, thus achieving goal *(2)* of BWCA. To achieve these two-part guarantees, we define the notion of perturbation resilience on a subset of the datapoints. All prior work has only studied perturbation resilience as it applies to the entire dataset. Informally, a subset $S' \subseteq S$ satisfies α-perturbation resilience if all points $v \in S'$ remain in the same optimal cluster under any α-perturbation to the input. We show that our algorithms return all optimal clusters from these locally stable regions. Most of our results also apply under the recently defined, weaker condition of α-*metric perturbation resilience* [Angelidakis et al., 2017], which states that the optimal solution cannot change under the metric closure of any α-perturbation. A summary of our results and techniques are as follows:

k-center under 2-perturbation resilience In Section 2.3, we show that *any* 2-approximation algorithm for k-center will always return the clusters satisfying 2-perturbation resilience. Therefore, since there are well-known 2-approximation algorithms for symmetric k-center, our analysis shows these will output the optimal clustering under 2-perturbation resilience. For asymmetric k-center, we give a new algorithm under 2-perturbation resilience, which works by first consider-

ing the "symmetrized set", or the points which demonstrate a rough symmetry. We show how to optimally cluster the symmetrized set, and then we show how to add back the highly asymmetric points into their correct clusters.

Hardness of symmetric k-center under $(2-\delta)$-perturbation resilience. In Section 2.3.3, we prove there is no polynomial time algorithm for symmetric k-center under $(2-\delta)$-perturbation resilience unless $NP = RP$, which shows that our perturbation resilience results are tight for both symmetric and asymmetric k-center. In particular, it implies that we have identified the exact moment ($\alpha = 2$) where the problem switches from efficiently computable to NP-hard, for both symmetric and asymmetric k-center. For this hardness result, we use a reduction from a variant of perfect dominating set. To show that this variant is itself hard, we construct a chain of parsimonious reductions (reductions which conserve the number of solutions) starting from 3-dimensional matching to perfect dominating set.

Asymmetric k-center In Section 2.5, we build off our result from Section 2.3 by applying the asymmetric k-center result to the local perturbation resilience and metric perturbation resilience settings. In the worst case, the asymmetric k-center problem admits an $O(\log^* n)$ approximation algorithm due to Vishwanathan [1996], which is tight [Chuzhoy et al., 2005]. We give new insights into this algorithm, which allow us to show a modification of the algorithm which outputs all optimal clusters from 2-perturbation resilient regions, while keeping the worst-case $O(\log^* n)$ guarantee overall. If the entire dataset satisfies 2-perturbation resilience, then our algorithm outputs the optimal clustering. 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. We show that each optimal center from a 2-perturbation resilient subset is a CCV and satisfies a separation property; we prove this by carefully constructing a 2-perturbation in which points from other clusters cannot be too close to the center without causing a contradiction. The structure allows us to modify the approximation algorithm of Vishwanathan to ensure that optimal clusters from perturbation resilient subsets are pulled out separately in the first phase. All of our guarantees hold under the weaker notion of metric perturbation resilience.

Efficient algorithms for k-center under $(3,\epsilon)$-perturbation resilience. In Section 2.6, we consider (α,ϵ)-perturbation resilience, which states that at most ϵn total points can swap into or out of each cluster under any α-perturbation. For symmetric k-center, we show that any 2-approximation algorithm will return the optimal clusters from $(3,\epsilon)$-perturbation resilient regions, assuming a mild lower bound on optimal cluster sizes, and for asymmetric k-center, we give an algorithm which outputs a clustering that is ϵ-close to the optimal clustering. Our main structural tool is showing that if any single point v is close to an optimal cluster other than its own, then $k-1$ centers achieve the optimal radius under a carefully constructed 3-perturbation. Any other point we add to the set of centers must create a clustering that is ϵ-close to the optimal clustering, and we show all of these sets cannot simultaneously be consistent with one another, thus causing a contradiction. A key concept in our analysis is defining the notion of a cluster-capturing center, which allows us to reason about which points can capture a cluster when its center is removed.

Our upper bound for asymmetric k-center under 2-PR and lower bound for symmetric k-center

under $(2-\delta)$-PR illustrate a surprising relationship between symmetric and asymmetric k-center instances under perturbation resilience. Unlike approximation ratio, for which symmetric k-center is easily solved to a factor of 2 but asymmetric k-center cannot be approximated to any constant factor, both symmetric and asymmetric k-center can be solved optimally under resilience to 2-perturbations. Overall, this is the first tight result quantifying the power of perturbation resilience for a canonical combinatorial optimization problem. A summary of our results can be found in Table 2.1.

Problem	**Guarantee**	α	**Metric**	**Local**	**Theorem**
Symmetric k-center under α-PR	$\mathcal{OPT}$	2	Yes	Yes	Theorem 2.5.1
Asymmetric k-center under α-PR	$\mathcal{OPT}$	2	Yes	Yes	Theorem 2.5.2
Symmetric k-center under (α, ϵ)-PR	$\mathcal{OPT}$	3	No	Yes	Theorem 2.6.8
Asymmetric k-center under (α, ϵ)-PR	ϵ-close	3	No	No	Theorem 2.6.17

Table 2.1: Our results over all variants of k-center under perturbation resilience

2.1.2 Related work

Clustering. There are three classic 2-approximations for k-center from the 1980's [Gonzalez, 1985, Hochbaum and Shmoys, 1985, Dyer and Frieze, 1985], which are known to be tight [Hochbaum and Shmoys, 1985]. Asymmetric k-center proved to be a much harder problem. The first nontrivial result was an $O(\log^* n)$ approximation algorithm [Vishwanathan, 1996], and this was later improved to $O(\log^* k)$ [Archer, 2001]. This result was later proven to be asymptotically tight [Chuzhoy et al., 2005].

Perturbation resilience. Perturbation resilience was introduced by Bilu and Linial [2012], 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. [2014]). The study of clustering under perturbation resilience was initiated by Awasthi et al. [2012], 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 and Liang [2016], 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.

Recently, Angelidakis et al. [2017] gave algorithms for center-based clustering (including k-median, k-means, and k-center) under 2-perturbation resilience, and defined the more general notion of metric perturbation resilience, although their algorithm does not extend to the (α, ϵ)-perturbation resilience or local perturbation resilience settings. Cohen-Addad and Schwiegelshohn [2017] showed that local search outputs the optimal k-median, k-means, and k-center solution when the data satisfies a stronger variant of 3-perturbation resilience, in which both the optimal clustering and optimal centers are not allowed to change under any 3-perturbation. Perturbation re-

silience has also been applied to other problems, such as min multiway cut, the traveling salesman problem, finding Nash equilibria, metric labeling, and facility location [Makarychev et al., 2014, Mihalák et al., 2011, Balcan and Braverman, 2017, Lang et al., 2017, Manthey and Tijink, 2018].

Subsequent work. Vijayaraghavan et al. [2017] study k-means under additive perturbation resilience, in which the optimal solution cannot change under additive perturbations to the input distances. Deshpande et al. [2018] gave an algorithm for Euclidean k-means under perturbation resilience which runs in time linear in n and the dimension d, and exponentially in k and $\frac{1}{\alpha-1}$. Chekuri and Gupta [2018] showed the natural LP relaxation of k-center and asymmetric k-center is integral for 2-perturbation resilient instances. They also define a new model of perturbation resilience for clustering with outliers, and they show the algorithm of Angelidakis et al. [2017] exactly solves clustering with outliers under 2-perturbation resilience, and they further show the natural LP relaxation for k-center with outliers is integral for 2-perturbation resilient instances. Their algorithms have the desirable property that either they output the optimal solution, or they guarantee the input did not satisfy 2-perturbation resilience (but note this is not the same thing as determining whether or not a given instance satisfies perturbation resilience).

Other stability notions. A related notion, approximation stability [Balcan et al., 2013a], states that any $(1+\alpha)$-approximation to the objective must be ϵ-close to the target clustering. There are several positive results for k-means, k-median [Balcan et al., 2013a, 2009, Gupta et al., 2014], and min-sum [Balcan et al., 2013a, Balcan and Braverman, 2009, Voevodski et al., 2011] under approximation stability. Ostrovsky et al. [2012] show how to efficiently cluster instances in which the k-means clustering cost is much lower than the $(k-1)$-means cost. Kumar and Kannan [2010] give an efficient clustering algorithm for instances in which the projection of any point onto the line between its cluster center to any other cluster center is a large additive factor closer to its own center than the other center. This result was later improved along multiple axes by Awasthi and Sheffet [2012]. There are many other works that show positive results for different natural notions of stability in various settings [Arora et al., 2012, Awasthi et al., 2010, Gupta et al., 2014, Hardt and Roth, 2013, Kumar and Kannan, 2010, Kumar et al., 2004, Roughgarden, 2014].

2.2 Preliminaries and basic properties

A clustering instance (S, d) consists of a set S of n points, a distance function $d : S \times S \to \mathbb{R}_{\geq 0}$, and an integer k. 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-center objective is to find a set of points $X = \{x_1, \ldots, x_k\} \subseteq S$ called *centers* to minimize $\max_{v \in S} d(v, X)$. 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 this chapter, we denote the clustering with minimum cost by $\mathcal{OPT} = \{C_1, \ldots, C_k\}$, we denote the radius of $\mathcal{OPT}$ by r^*, and we denote the optimal centers by $c_1, \ldots, c_k$, where c_i is the center of C_i for all $1 \leq i \leq k$. We use $B_r(c)$ to denote a ball of radius r centered at point r.

Some of our results assume distance functions which are metrics, and some of our results assume *asymmetric* distance functions. A distance function d is a *metric* if

1. for all u, v, $d(u, v) \geq 0$,

2. for all u, v, $d(u, v) = 0$ if and only if $u = v$,

3. for all u, v, w, $d(u, w) \leq d(u, v) + d(v, w)$, and

4. for all u, v, $d(u, v) = d(v, u)$.

An *asymmetric* distance function satisfies *(1)*, *(2)*, and *(3)*, but not *(4)*.

Now we formally define *perturbation resilience*, a notion introduced by Bilu and Linial [2012]. d' is called an α-perturbation of the distance function d, if for all $u, v \in S$, $d(u, v) \leq d'(u, v) \leq \alpha d(u, v)$. [1]

Definition 2.2.1. *(Perturbation resilience) A clustering instance (S, d) satisfies* α-perturbation resilience *(α-PR) if for any α-perturbation d' of d, the optimal clustering $\mathcal{C}'$ under d' is unique and equal to $\mathcal{OPT}$.*

Note that the optimal *centers* might change under an α-perturbation, but the optimal *clustering* must stay the same. We also consider a relaxed variant of α-perturbation resilience, called (α, ϵ) -perturbation resilience, that allows a small change in the optimal clustering when distances are perturbed. We say that two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are ϵ-close if $\min_\sigma \sum_{i=1}^{k} \left| C_i \setminus C'_{\sigma(i)} \right| \leq \epsilon n$, where σ is a permutation on $[k]$.

Definition 2.2.2. *((α, ϵ)-perturbation resilience) A clustering instance (S, d) satisfies (α, ϵ)- perturbation resilience if for any α-perturbation d' of d, any optimal clustering $\mathcal{C}'$ under d' is ϵ-close to $\mathcal{OPT}$.*

In Definitions 2.2.1 and 2.2.2, we do not assume that the α-perturbations satisfy the triangle inequality. Angelidakis et al. [2017] recently studied the weaker definition in which the α-perturbations must satisfy the triangle inequality, called *metric perturbation resilience*. We can update these definitions accordingly. For symmetric clustering objectives, α-metric perturbations are restricted to metrics. For asymmetric clustering objectives, the α-metric perturbations must satisfy the directed triangle inequality.

Definition 2.2.3. *(Metric perturbation resilience) A clustering instance (S, d) satisfies* α-metric perturbation resilience *(α-MPR) if for any α-metric perturbation d' of d, the optimal clustering $\mathcal{C}'$ under d' is unique and equal to $\mathcal{OPT}$.*

In our arguments, we will sometimes convert a non-metric perturbation d' into a metric perturbation by taking the *metric completion* d'' of d' (also referred to as the *shortest-path metric* on d') by setting the distances in d'' as the length of the shortest path on the graph whose edges are the lengths in d'. Note that for all u, v, we have $d(u, v) \leq d'(u, v)$ since d was originally a metric.

[1] We only consider perturbations in which the distances increase because WLOG we can scale the distances to simulate decreasing distances.

2.2.1 Local Perturbation Resilience

Now we define perturbation resilience for an optimal cluster rather than the entire dataset. All prior work has considered perturbation resilience with respect to the entire dataset.

Definition 2.2.4. *(Local perturbation resilience) Given a clustering instance (S, d) with optimal clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, an optimal cluster C_i satisfies α-perturbation resilience (α-PR) if for any α-perturbation d' of d, the optimal clustering $\mathcal{C}'$ under d' is unique and contains C_i.*

As a sanity check, we show that a clustering is perturbation resilient if and only if every optimal cluster satisfies perturbation resilience.

Fact 2.2.5. *A clustering instance (S, d) satisfies α-PR if and only if each optimal cluster satisfies α-PR.*

Proof. Given a clustering instance (S, d), the forward direction follows by definition: assume (S, d) satisfies α-PR, and given an optimal cluster C_i, then for each α-perturbation d', the optimal clustering stays the same under d', therefore C_i is contained in the optimal clustering under d'. Now we prove the reverse direction. Given a clustering instance with optimal clustering $\mathcal{C}$, and given an α-perturbation d', let the optimal clustering under d' be $\mathcal{C}'$. For each $C_i \in \mathcal{C}$, by assumption, C_i satisfies α-PR, so $C_i \in \mathcal{C}'$. Therefore $\mathcal{C} = \mathcal{C}'$. □

Next we define the local version of (α, ϵ)-PR.

Definition 2.2.6. *(Local (α, ϵ)-perturbation resilience) Given a clustering instance (S, d) with optimal clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, an optimal cluster C_i satisfies (α, ϵ)-PR if for any α-perturbation d' of d, the optimal clustering $\mathcal{C}'$ under d' contains a cluster C_i' which is ϵ-close to C_i.*

In Sections 2.5 and 2.6, we will consider a slightly stronger notion of local perturbation resilience. Informally, an optimal cluster satisfies α-strong perturbation resilience if it is α-PR, and all nearby optimal clusters are also α-PR. We will sometimes be able to prove guarantees for clusters satisfying strong perturbation resilience which are not true under standard perturbation resilience.

Definition 2.2.7. *(Strong perturbation resilience) Given a clustering instance (S, d) with optimal clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, an optimal cluster C_i satisfies α-strong perturbation resilience (α-SPR) if for each j such that there exists $u \in C_i$, $v \in C_j$, and $d(u, v) \leq r^*$, then C_j is α-PR (any cluster that is close to C_i must be α-PR).*

To conclude this section, we state a lemma for asymmetric (and symmetric) k-center which allows us to reason about a specific class of α-perturbations which will be important throughout the chapter. We give two versions of the lemma, each of which will be useful in different sections of the chapter.

Lemma 2.2.8. *Given a clustering instance (S, d) and $\alpha \geq 1$,* (a) *assume we have an α-perturbation d' of d with the following property: for all p, q, if $d(p, q) \geq r^*$ then $d'(p, q) \geq \alpha r^*$. Then the optimal cost under d' is αr^*.*

(b) *assume we have an α-perturbation d' of d with the following property: for all u, v, either $d'(u, v) = \min(\alpha r^*, \alpha d(u, v))$ or $d'(u, v) = \alpha d(u, v)$. Then the optimal cost under d' is αr^*.*

Proof. (a) Assume there exists a set of centers $C' = \{c'_1, \ldots, c'_k\}$ whose k-center cost under d' is $< \alpha r^*$. Then for all i and $s \in \text{Vor}_{C',d'}(c'_i)$, $d'(c'_i, s) < \alpha r^*$, implying $d(c'_i, s) < r^*$ by construction. It follows that the k-center cost of C' under d is r^*, which is a contradiction. Therefore, the optimal cost under d' must be αr^*.

(b) Given u, v such that $d(u, v) \geq r^*$, then $d'(u, v) \geq \alpha r^*$ by construction. Now the proof follows from part (a). □

2.3 k-center under α-perturbation resilience

In this section, we provide efficient algorithms for finding the optimal clustering for symmetric and asymmetric instances of k-center under 2-perturbation resilience. Our results directly improve on the result of Balcan and Liang [2016] for symmetric k-center under $(1 + \sqrt{2})$-perturbation resilience. We also show that it is NP-hard to recover $\mathcal{OPT}$ even for symmetric k-center instance under $(2 - \delta)$-perturbation resilience. As an immediate consequence, our results are tight for both symmetric and asymmetric k-center instances. This is the first problem for which the exact value of perturbation resilience is found ($\alpha = 2$), where the problem switches from efficiently computable to NP-hard.

First, we show that any α-approximation algorithm returns the optimal solution for α perturbation resilient instances. An immediate consequence is an algorithm for symmetric k-center under 2-perturbation resilience. Next, we provide a novel algorithm for asymmetric k-center under 2-perturbation resilience. Finally, we show hardness of k-center under $(2 - \delta)$-PR.

2.3.1 α-approximations are optimal under α-PR

The following theorem shows that any α-approximation algorithm for k-center will return the optimal solution on clustering instances that are α-perturbation resilient.

Theorem 2.3.1. *Given a clustering instance (S, d) satisfying α-perturbation resilience for asymmetric k-center, and a set C of k centers which is an α-approximation, i.e., $\forall p \in S$, $\exists c \in C$ s.t. $d(c, p) \leq \alpha r^*$, then the Voronoi partition induced by C is the optimal clustering.*

Proof. For a point $p \in S$, let $c(p) := \text{argmin}_{c \in C} d(c, p)$, the closest center in C to p. The idea is to construct an α-perturbation in which C is the optimal solution by increasing all distances except between p and $c(p)$, for all p. Then the theorem will follow by using the definition of perturbation resilience.

By assumption, $\forall p \in S$, $d(c(p), p) \leq \alpha r^*$. Create a perturbation d' as follows. Increase all distances by a factor of α, except for all $p \in S$, set $d'(c(p), p) = \min(\alpha d(c(p), p), \alpha r^*)$ (recall in Definition 2.2.1, the perturbation need not satisfy the triangle inequality). Then no distances

were increased by more than a factor of α. And since we had that $d(c(p), p) \leq \alpha r^*$, no distances decrease either. Therefore, d' is an α-perturbation of d. By Lemma 2.2.8, the optimal cost for d' is αr^*. Also, C achieves cost $\leq \alpha r^*$ by construction, so C is an optimal set of centers under d'. Then by α-perturbation resilience, the Voronoi partition induced by C under d' is the optimal clustering.

Finally, we show the Voronoi partition of C under d is the same as the Voronoi partition of C under d'. Given $p \in S$ whose closest point in C is $c(p)$ under d, then under d', all distances from p to $C \setminus \{c(p)\}$ increased by exactly α, and $d(p, c(p))$ increased by $\leq \alpha$. Therefore, the closest point in C to p under d' is still $c(p)$. □

2.3.2 k-center under 2-PR

An immediate consequence of Theorem 2.3.1 is that we have an exact algorithm for symmetric k-center under 2-perturbation resilience by running a simple 2-approximation algorithm (e.g., [Gonzalez, 1985, Hochbaum and Shmoys, 1985, Dyer and Frieze, 1985]). However, Theorem 2.3.1 only gives an algorithm for asymmetric k-center under $O(\log^*(k))$-perturbation resilience. Next, we show it is possible to substantially improve the latter result.

Asymmetric k-center under 2-PR

One of the challenges involved in dealing with asymmetric k-center instances is the fact that even though for all $p \in C_i$, $d(c_i, p) \leq r^*$, the reverse distance, $d(p, c_i)$, might be arbitrarily large. Such points for which $d(p, c_i) \gg r^*$ pose a challenge to the structure of the clusters, as they can be very close to points or even centers of other clusters. To deal with this challenge, we first define a set of "good" points, A, such that $A = \{p \mid \forall q, d(q, p) \leq r^* \implies d(p, q) \leq r^*\}$. [2] Intuitively speaking, these points behave similarly to a set of points with symmetric distances up to a distance r^*. To explore this, we define a desirable property of A with respect to the optimal clustering.

Definition 2.3.2. *A is said to* respect the structure of *$\mathcal{OPT}$ if*

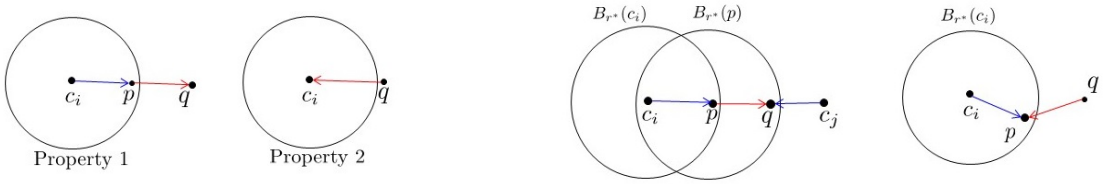
(1) $c_i \in A$ for all $i \in [k]$, and

(2) for all $p \in S \setminus A$, if $A(p) := \arg\min_{q \in A} d(q, p) \in C_i$, then $p \in C_i$.

For all i, define $C'_i = C_i \cap A$ (which is in fact the optimal clustering of A). Satisfying Definition 2.3.2 implies that if we can optimally cluster A, then we can optimally cluster the entire instance (formalized in Theorem 2.3.5). Thus our goal is to show that A does indeed respect the structure of $\mathcal{OPT}$, and to show how to return $C'_1, \ldots, C'_k$.

Intuitively, A is similar to a symmetric 2-perturbation resilient clustering instance. However, some structure is no longer there, for instance, a point p may be at distance $\leq 2r^*$ from every point in a different cluster, which is not true for 2-perturbation resilient instances. This implies we cannot simply run a 2-approximation algorithm on the set A, as we did in the previous section. However, we show that the remaining structural properties are sufficient to optimally cluster A. To this end, we define two properties and show how they lead to an algorithm that returns $C'_1, \ldots, C'_k$, and help us prove that A respects the structure of $\mathcal{OPT}$.

[2] A "good" point is referred to as a *center-capturing vertex* in other works, e.g., Vishwanathan [1996]. We formally define this notion in Section 2.5.



(a) Properties of 2-perturbation resilience

(b) Demonstrating the correctness of Algorithm 1

Figure 2.1: Properties of a 2-perturbation resilient instance of aymmetric k-center that are used for clustering.

The first of these properties requires each point to be closer to its center than any point in another cluster.

Property (1): For all $p \in C'_i$ and $q \in C'_{j \neq i}$, $d(c_i, p) < d(q, p)$.

The second property requires that any point within distance r^* of a cluster center belongs to that cluster.

Property (2): For all $i \neq j$ and $q \in C_j$, $d(q, c_i) > r^*$ (see Figure 2.1). [3]

Let us illustrate how these properties allow us to optimally cluster A. [4] Consider a ball of radius r^* around a center c_i. By Property 2, such a ball exactly captures C'_i. Furthermore, by Property 1, any point in this ball is closer to the center than to points outside of the ball. Is this true for a ball of radius r^* around a general point p? Not necessarily. If this ball contains a point $q \in C'_j$ from a different cluster, then q will be closer to a point outside the ball than to p (namely, c_j, which is guaranteed to be outside of the ball by Property 2). This allows us to determine that the center of such a ball must not be an optimal center.

This structure motivates our Algorithm 1 for asymmetric k-center under 2-perturbation resilience. At a high level, we start by constructing the set A (which can be done easily in polynomial time). Then we create the set of all balls of radius r^* around all points in A (if r^* is not known, we can use a guess-and-check wrapper). Next, we prune this set by throwing out any ball that contains a point farther from its center than to a point outside the ball. We also throw out any ball that is a subset of another one. Our claim is that the remaining balls are exactly $C'_1, \ldots, C'_k$. Finally, we add the points in $S \setminus A$ to their closest point in A.

Formal details of our analysis

Lemma 2.3.3. *Properties 1 and 2 hold for asymmetric k-center instances satisfying 2-perturbation resilience.*

[3] Property (1) first appeared in the work of Awasthi et al. [2012], for symmetric clustering instances. A weaker variation of Property (2) was introduced by Balcan and Liang [2016], which showed that in $1 + \sqrt{2}$-perturbation resilient instances for any cluster C_i with radius r_i, $B_{r_i}(c_i) = C_i$. Our Property (2) shows that this is true *for a universal radius, r^**, even for 2-perturbation resilient instances, *and* even for asymmetric instances.

[4] Other algorithms work, such as single linkage with dynamic programming at the end to find the minimum cost pruning of k clusters. However, our algorithm is able to recognize optimal clusters *locally* (without a complete view of the point set).

Algorithm 1 ASYMMETRIC k-CENTER ALGORITHM UNDER 2-PR

Input: Asymmetric k-center instance (S, d), distance r^* (or try all possible candidates)

Create symmetric set

- Build set $A = \{p \mid \forall q, d(q,p) \leq r^* \implies d(p,q) \leq r^*\}$

Create candidate balls

- $\forall c \in A$, construct $G_c = B_{r^*}(c)$ (the ball of radius r^* around c).

Prune balls

- $\forall G_c$, if $\exists p \in G_c$, $q \notin G_c$ s.t. $d(q,p) < d(c,p)$, then throw out G_c.
- $\forall p, q$ s.t. $G_p \subseteq G_q$, throw out G_p.

Insert remaining points

- $\forall p \notin A$, add p to G_q, where $q = \arg\min_{s \in A} d(s,p)$.

Output: sets $G_1, \ldots, G_k$

Proof. Property 1: Assume false, $d(q,p) \leq d(c_i,p)$. The idea will be that since q is in A, it is close to its own center, so we can construct a perturbation in which q replaces its center c_j. Then p will join q's cluster, causing a contradiction. Construct the following d':

$$d'(s,t) = \begin{cases} \min(2r^*, 2d(s,t)) & \text{if } s = q,\ t \in C_j \cup \{p\} \\ 2d(s,t) & \text{otherwise.} \end{cases}$$

This is a 2-perturbation because $d(q, C_j \cup \{p\}) \leq 2r^*$. Then by Lemma 2.2.8, the optimal cost is $2r^*$. The set of centers $\{c_\ell\}_{i=1}^k \setminus \{c_j\} \cup \{q\}$ achieves the optimal cost, since q is distance $2r^*$ from C_j, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $2r^*$). Then for all c_ℓ, $d'(q,p) \leq d'(c_i,p) \leq d'(c_\ell,p)$. And since $q \in A$, $d(q,c_j) \leq r^*$ so $d(q,C_j) \leq 2r^*$. Then we can construct a 2-perturbation in which q becomes the center of C_j, and then q is the best center for p, so we have a contradiction.

Property 2: Assume on the contrary that there exists $q \in C_j$, $i \neq j$ such that $d(q,c_i) \leq r^*$. Now we will define a d' in which q can become a center for C_i.

$$d'(s,t) = \begin{cases} \min(2r^*, 2d(s,t)) & \text{if } s = q,\ t \in C_i \\ 2d(s,t) & \text{otherwise.} \end{cases}$$

This is a 2-perturbation because $d(q, C_i) \leq 2r^*$. Then by Lemma 2.2.8, the optimal cost is $2r^*$. The set of centers $\{c_\ell\}_{i=1}^k \setminus \{c_i\} \cup \{q\}$ achieves the optimal cost, since q is distance $2r^*$ from C_i, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $2r^*$). But the clustering with centers $\{c_\ell\}_{i=1}^k \setminus \{c_i\} \cup \{q\}$ is different from $\mathcal{OPT}$, since (at the very least) q and c_i are in different clusters. This contradicts 2-perturbation resilience. □

Lemma 2.3.4. *The set A respects the structure of $\mathcal{OPT}$.*

Proof. From Lemma 2.3.3, we can use Property 2 in our analysis. First we show that $c_i \in A$ for all $i \in [k]$. Given c_i, $\forall p \in C_i$, then $d(c_i, p) \leq r^*$ by definition of $\mathcal{OPT}$. $\forall q \notin C_i$, then by Property 2, $d(q, c_i) > r^*$. It follows that for any point $p \in S$, it cannot be the case that $d(p, c_i) \leq r^*$ and $d(c_i, p) > r^*$. Therefore, $c_i \in A$.

Now we show that for all $p \in S \setminus A$, if $A(p) \in C_i$, then $p \in C_i$. Given $p \in S \setminus A$, let $p \in C_i$ and assume towards contradiction that $q = A(p) \in C_j$ for some $i \neq j$. We will construct a 2-perturbation d' in which q replaces c_j as the center for C_j and p switches from C_i to C_j, causing a contradiction. We construct d' as follows. All distances are increased by a factor of 2 except for $d(q, p)$ and $d(q, q')$ for all $q' \in C_j$. These distances are increased by a factor of 2 up to $2r^*$. Formally,

$$d'(s,t) = \begin{cases} \min(2r^*, 2d(s,t)) & \text{if } s = q,\, t \in C_j \cup \{p\} \\ 2d(s,t) & \text{otherwise.} \end{cases}$$

This is a 2-perturbation because $d(q, C_j) \leq 2r^*$. Then by Lemma 2.2.8, the optimal cost is $2r^*$. The set of centers $\{c_\ell\}_{i=1}^k \setminus \{c_j\} \cup \{q\}$ achieves the optimal cost, since q is distance $2r^*$ from C_j, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $2r^*$). But consider the point p. Since all centers are in A and q is the closest point to p in A, then q is the center for p under d'. Therefore, the optimal clustering under d' is different from $\mathcal{OPT}$, so we have a contradiction. □

Now we are ready to show Algorithm 1 returns the optimal clustering.

Theorem 2.3.5. *Algorithm 1 returns the exact solution for asymmetric k-center under 2-perturbation resilience.*

Proof. First we must show that after step 1, the remaining sets are exactly $C_1', \ldots, C_k' = C_1 \cap A, \ldots, C_k \cap A$. We prove this in three steps: the sets G_{c_i} correspond to C_i', these sets are not thrown out in steps 1 and 1, and all other sets are thrown out in steps 1 and 1. Because of Lemma 2.3.3, we can use Properties 1 and 2.

For all i, $G_{c_i} = C_i'$: From Lemma 2.3.4, all centers are in A, so G_{c_i} will be created in step 2. For all $p \in C_i$, $d(c_i, p) \leq r^*$. For all $q \notin C_i'$, then by Property 2, $d(q, c_i) > r^*$ (and since $c_i, q \in A$, $d(c_i, q) > r^*$ as well). For all i, G_{c_i} is not thrown out in step 1: Given $s \in G_{c_i}$ and $t \notin G_{c_i}$. Then $s \in C_i'$ and $t \in C_j'$ for $j \neq i$. If $d(t, s) < d(c_i, s)$, then we get a contradiction from Property 1. For all non-centers p, G_p is thrown out in step 1 or 1: From the previous paragraph, $G_{c_i} = C_i'$. If $G_p \subseteq G_{c_i}$, then G_p will be thrown out in step 1 (if $G_p = G_{c_i}$, it does not matter which set we keep, so WLOG say that we keep G_{c_i}). Then if G_p is not thrown out in step 1, $\exists s \in G_p \cap C_j'$, $j \neq i$. If $s = c_j$, then $d(p, c_j) \leq r^*$ and we get a contradiction from Property 2. So, we can assume s is a non-center (and that $c_j \notin G_p$). But $d(c_j, s) < d(p, s)$ from Property 1, and therefore G_p will be thrown out in step 1. Thus, the remaining sets after step 1 are exactly $C_1', \ldots, C_k'$.

Finally, by Lemma 2.3.4, for each $p \in C_i \setminus A$, $A(p) \in C_i$, so p will be added to G_{c_i}. Therefore, the final output is $C_1, \ldots, C_k$. □

2.3.3 Hardness of k-center under perturbation resilience

In this section, we show NP-hardness for k-center under $(2-\delta)$-perturbation resilience. We show that if symmetric k-center under $(2-\delta)$-perturbation resilience [5] can be solved in polynomial time, then $NP = RP$, even under the condition that the optimal clusters are all size $\geq \frac{n}{2k}$. Because symmetric k-center is a special case of asymmetric k-center, we have the same hardness results for asymmetric k-center. This proves Theorem 2.3.5 is tight with respect to the level of perturbation resilience assumed.

Theorem 2.3.6. *There is no polynomial time algorithm for finding the optimal k-center clustering under $(2-\delta)$-perturbation resilience, 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. Below, we formally define this problem and all intermediate problems. Part of our reduction is based off of the proof of Ben-David and Reyzin [2012], who showed a reduction from a variant of dominating set to the weaker problem of clustering under $(2-\delta)$-center proximity [6] We use four NP-hard problems in a chain of reductions. Here, we define all of these problems up front. We introduce the "balanced" variants of two existing problems.

Definition 2.3.7 (3-Dimensional Matching (3DM))**.** *We are given three disjoint sets X_1, X_2, and X_3 each of size m, and a set T such that $t \in T$ is a triple $t = (x_1, x_2, x_3)$ where $x_1 \in X_1$, $x_2 \in X_2$, and $x_3 \in X_3$. The problem is to find a set $M \subseteq T$ of size m which exactly hits all the elements in $X_1 \cup X_2 \cup X_3$. In other words, for all pairs $(x_1, x_2, x_3), (y_1, y_2, y_3) \in M$, it is the case that $x_1 \neq y_1$, $x_2 \neq y_2$, and $x_3 \neq y_3$.*

Definition 2.3.8 (Balanced-3-Dimensional Matching (B3DM))**.** *This is the 3DM problem (X_1, X_2, X_3, T) with the additional constraint that $2m \leq |T| \leq 3m$, where $|X_1| = |X_2| = |X_3| = m$.*

Definition 2.3.9 (Perfect Dominating Set (PDS))**.** *Given a graph $G = (V, E)$ and an integer k, the problem is to find a set of vertices $D \subseteq V$ of size k such that for all $v \in V \setminus D$, there exists* exactly one *$d \in D$ such that $(v, d) \in E$.*

Definition 2.3.10 (Balanced-Perfect-Dominating Set (BPDS))**.** *This is the PDS problem (G, k) with the additional assumption that if the graph has n vertices and a dominating set of size k exists, then each vertex in the dominating set hits at least $\frac{n}{2k}$ vertices.*

Additionally, each problem has an "Unambiguous" variant, which is the added constraint that the problem has at most one solution. Valiant and Vazirani showed that Unambiguous-3SAT is hard unless $NP = RP$ [Valiant and Vazirani, 1986]. To show the Unambiguous version of another problem is hard, one must establish a parsimonious reduction from Unambiguous-3SAT to that problem. A parsimonious reduction is one that conserves the number of solutions. For two

[5] In fact, our result holds even under the strictly stronger notion of *approximation stability* [Balcan et al., 2013a].

[6] 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.

problems A and B, we denote $A \leq_{par} B$ to mean there is a reduction from A to B that is parsimonious *and* polynomial. Some common reductions involve 1-to-1 mappings which are easy to verify parsimony, but many other common reductions are not parsimonious. For instance, the standard reduction from 3SAT to 3DM is not parsimonious [Kleinberg and Tardos, 2006], yet there is a more roundabout series of reductions which all use 1-to-1 mappings, and are therefore easy to verify parsimony. In order to prove Theorem 2.3.6, we start with the claim that Unambiguous-BPDS is hard unless $NP = RP$. We use a parsimonious series of reductions from 3SAT to B3DM to BPDS. All of these reductions are from prior work, yet we verify parsimony and balancedness.

Lemma 2.3.11. *There is no polynomial time algorithm for Unambiguous-BPDS unless $NP = RP$.*

Proof. We give a parsimonious series of reductions from 3SAT to B3DM to BPDS. Then it follows from the result of Valiant and Vazirani [1986] that there is no polynomial time algorithm for Unambiguous-BPDS unless $NP = RP$.

To show that B3DM is NP-hard, we use the reduction of Dyer and Frieze [1986] who showed that Planar-3DM is NP-hard. While planarity is not important for the purpose of our problems, their reduction from 3SAT has two other nice properties that we crucially use. First, the reduction is parsimonious, as pointed out by Hunt III et al. [1998]. Second, given their 3DM instance X_1, X_2, X_3, T, each element in $X_1 \cup X_2 \cup X_3$ appears in either two or three tuples in T. (Dyer and Frieze [1986] mention this observation just before their Theorem 2.3.) From this, it follows that $2m \leq |T| \leq 3m$, and so their reduction proves that B3DM is NP-hard via a parsimonious reduction from 3SAT.

Next, we reduce B3DM to BPDS using a reduction similar to the reduction in Ben-David and Reyzin [2012]. Their reduction maps every element X_1, X_2, X_3, T to a vertex in V, and adds one extra vertex v to V. There is an edge from each element $(x_1, x_2, x_3) \in T$ to the corresponding elements $x_1 \in X_1$, $x_2 \in X_2$, and $x_3 \in X_3$. Furthermore, there is an edge from v to every element in T. Ben-David and Reyzin [2012] show that if the 3DM instance is a YES instance with matching $M \subseteq T$ then the minimum dominating set is $v \cup M$. Then, this dominating set is size $m + 1$. If we start with B3DM, our graph has $|X_1| + |X_2| + |X_3| + |T| + 1 \leq 6m + 1$ vertices since $|T| \leq 3m$. Given $t \in M$, t hits 3 nodes in the graph, and $\frac{n}{2(m+1)} \leq \frac{6m+1}{2m+2} \leq 3$. Furthermore, v hits $|T| - m \geq 2m - m = m$ nodes, and $\frac{6m+1}{2m+2} \leq m$ when $m \geq 3$. Therefore, the resulting instance is a BPDS instance.

Now we have verified that there exists a parsimonious reduction 3SAT $\leq_{par}$ BPDS, so it follows that there is no polynomial time algorithm for Unambiguous-BPDS unless $NP = RP$. □

Now we can prove Theorem 2.3.6 by giving a reduction from Unambiguous-BPDS to k-center clustering under $(2 - \delta)$-perturbation resilience, where all clusters are size $\geq \frac{n}{2k}$. We use the same reduction as in [Ben-David and Reyzin, 2012], but we must verify that the resulting instance is $(2 - \delta)$-perturbation resilient, which requires the unambiguity.

Proof of Theorem 2.3.6. From Lemma 2.3.11, Unambiguous-BPDS is NP-hard unless $NP = RP$. Now for all $\delta > 0$, we reduce from Unambiguous-BPDS to k-center clustering and show the resulting instance has all cluster sizes $\geq \frac{n}{2k}$ and satisfies $(2 - \delta)$-perturbation resilience.

Given an instance of Unambiguous-BPDS, for every $v \in V$, create a point $v \in S$ in the clustering instance. For every edge $(u, v) \in E$, let $d(u, v) = 1$, otherwise let $d(u, v) = 2$. Since all distances are either 1 or 2, the triangle inequality is trivially satisfied. Then a k-center solution of cost 1 exists if and only if there exists a dominating set of size k.

Since each vertex in the dominating set hit at least $\frac{n}{2k}$ vertices, the resulting clusters will be size at least $\frac{n}{2k} + 1$. Additionally, if there exists a dominating set of size k, then the corresponding optimal k-center clustering has cost 1. Because this dominating set is perfect and unique, any other clustering has cost 2. It follows that the k-center instance is $(2 - \delta)$-perturbation resilient. □

2.4 k-center under metric perturbation resilience

In this section, we extend the results from Section 2.3 to the metric perturbation resilience setting. We first give a generalization of Lemma 2.2.8 to show that it can be extended to metric perturbation resilience. Then we show how this immediately leads to corollaries of Theorem 2.3.1 and Theorem 2.3.5 extended to the metric perturbation resilience setting.

Recall that in the proofs from the previous section, we created α-perturbations d' by increasing all distances by α, except a few distances $d(u, v) \leq \alpha r^*$ which we increased to $\min(\alpha d(u, v), \alpha r^*)$. In this specific type of α-perturbation, we used the crucial property that the optimal clustering has cost αr^* (Lemma 2.2.8). 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' (recall the metric completion of d' is defined as the shortest path metric on d'). In the following lemma, we show that Lemma 2.2.8 remains true after taking the metric completion of the perturbation.

Lemma 2.4.1. *Given $\alpha \geq 1$ and an asymmetric k-center clustering instance $\mathcal{V} = (V, d, k)$ with optimal radius r^*, let d'' denote an α-perturbation such that for all u, v, either $d''(u, v) = \min(\alpha r^*, \alpha d(u, v))$ or $d''(u, v) = \alpha d(u, v)$. Let d' denote the metric completion of d''. Then d' is an α-metric perturbation of d, and the optimal cost under d' is αr^*.*

Proof. By construction, $d'(u, v) \leq d''(u, v) \leq \alpha d(u, v)$. Since d satisfies the triangle inequality, we have that $d(u, v) \leq d'(u, v)$, so d' is a valid α-metric perturbation of d.

Now given u, v such that $d(u, v) \geq r^*$, we will prove that $d'(u, v) \geq \alpha r^*$. By construction, $d''(u, v) \geq \alpha r^*$. Then since d' is the metric completion of d'', there exists a path $u = u_0$–u_1–$\cdots$–u_{s-1}–$u_s = v$ such that $d'(u, v) = \sum_{i=0}^{s-1} d'(u_i, u_{i+1})$ and for all $0 \leq i \leq s - 1$, $d'(u_i, u_{i+1}) = d''(u_i, u_{i+1})$.

Case 1: there exists an i such that $d''(u_i, u_{i+1}) \geq \alpha r^*$. Then $d'(u, v) \geq \alpha r^*$ and we are done.

Case 2: for all $0 \leq i \leq s - 1$, $d''(u_i, u_{i+1}) < \alpha r^*$. Then by construction, $d'(u_i, u_{i+1}) = d''(u_i, u_{i+1}) = \alpha d(u_i, u_{i+1})$, and so

$$d'(u, v) = \sum_{i=0}^{s-1} d'(u_i, u_{i+1}) = \alpha \sum_{i=0}^{s-1} d(u_i, u_{i+1}) \geq \alpha d(u, v) \geq \alpha r^*.$$

We have proven that for all u, v, if $d(u, v) \geq r^*$, then $d'(u, v) \geq \alpha r^*$. Assume there exists a set of centers $C' = \{c'_1, \ldots, c'_k\}$ whose k-center cost under d' is $< \alpha r^*$. Then for all i and

$s \in \text{Vor}_{C',d'}(c'_i)$, $d'(c'_i, s) < \alpha r^*$, implying $d(c'_i, s) < r^*$ by construction. It follows that the k-center cost of C' under d is r^*, which is a contradiction. Therefore, the optimal cost under d' must be αr^*. □

Recall that metric perturbation resilience states that the optimal solution does not change under any metric perturbation to the input distances. In the proofs of Theorems 2.3.1 and 2.3.5, the only perturbations constructed were the type as in Lemma 2.2.8. Since Lemma 2.4.1 shows this type of perturbation is indeed a metric, Theorems 2.3.1 and 2.3.5 are true even under metric perturbation resilience.

Corollary 2.4.2. *Given a clustering instance (S, d) satisfying α-metric perturbation resilience for asymmetric k-center, and a set C of k centers which is an α-approximation, i.e., $\forall p \in S$, $\exists c \in C$ s.t. $d(c, p) \leq \alpha r^*$, then the Voronoi partition induced by C is the optimal clustering.*

Corollary 2.4.3. *Algorithm 1 returns the exact solution for asymmetric k-center under 2- perturbation resilience.*

2.5 k-center under local perturbation resilience

In this section, we further extend the results from Sections 2.3 and 2.4 to the local perturbation resilience setting. First we show that any α-approximation to k-center will return each optimal α-MPR cluster, i.e., Corollary 2.4.2 holds even in the local perturbation resilience setting. Then for asymmetric k-center, we show that a natural modification to the $O(\log^* n)$ approximation algorithm of Vishwanathan [1996] leads to an algorithm that maintains its performance in the worst case, while exactly returning each optimal cluster located within a 2-MPR region of the dataset. This generalizes Corollary 2.4.3.

2.5.1 Symmetric k-center

In section 2.3, we showed that any α-approximation algorithm for k-center returns the optimal solution for instances satisfying α-perturbation resilience (and this was generalized to metric perturbation resilience in the previous section). In this section, we extend this result to the local perturbation resilience setting. We show that any α-approximation will return each (local) α-MPR cluster. For example, if a clustering instance is half 2-perturbation resilient, running a 2-approximation algorithm will return the optimal clusters for half the dataset, and a 2-approximation for the other half.

Theorem 2.5.1. *Given an asymmetric k-center clustering instance (S, d), a set C of k centers which is an α-approximation, and a clustering $\mathcal{C}$ defined as the Voronio partition induced by C, then each α-MPR cluster is contained in $\mathcal{C}$.*

The proof is very similar to the proof of Theorem 2.3.1. The key difference is that we reason about each perturbation resilient cluster individually, rather than reasoning about the global structure of perturbation resilience.

Proof. Given an α-approximate solution $\mathcal{C}$ to a clustering instance (S, d), and given an α-MPR cluster C_i, we will create an α-perturbation as follows. Define

$$d''(v, \mathcal{C}(v)) = \min\{\alpha r^*, \alpha d(v, \mathcal{C}(v))\}, \text{For all } v \in S.$$

For all other points $u \in S$, set $d''(v, u) = \alpha d(v, u)$. Then by Lemma 2.4.1, the metric completion d' of d'' is an α-perturbation of d with optimal cost αr^*. By construction, the cost of $\mathcal{C}$ is $\leq \alpha r^*$ under d', therefore, $\mathcal{C}$ is an optimal clustering. Denote the set of centers of $\mathcal{C}$ by C. By definition of α-MPR, there exists $v_i \in C$ such that $\text{Vor}_C(v_i) = C_i$ in d'. Now, given $v \in C_i$, $\text{argmin}_{u \in C} d'(u, v) = v_i$, so by construction, $\text{argmin}_{u \in C} d(u, v) = v_i$. Therefore, $\text{Vor}_C(v_i) = C_i$, so $C_i \in \mathcal{C}$.

□

2.5.2 Asymmetric k-center

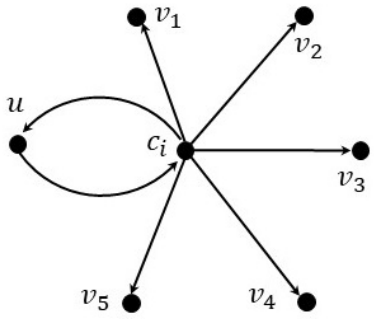
In Section 2.3, we gave an algorithm which outputs the optimal clustering for asymmetric k-center under 2-perturbation resilience (Algorithm 1 and Theorem 2.3.5), and we extended it to metric perturbation resilience in Section 2.4. In this section, we extend the result further to the local perturbation resilience setting, and we show how to add a worst-case guarantee of $O(\log^* n)$. Specifically, we give a new algorithm, which is a natural modification to the $O(\log^* n)$ approximation algorithm of Vishwanathan [1996], and show that it maintains the $O(\log^* n)$ guarantee in the worst case while returning each optimal perturbation resilient cluster in its own superset. As a consequence, if the entire clustering instance satisfies 2-metric perturbation resilience, then the output of our algorithm is the optimal clustering.

Theorem 2.5.2. *Given an asymmetric k-center clustering instance (S, d) of size n with optimal clustering $\{C_1, \ldots, C_k\}$, for each 2-MPR cluster C_i, there exists a cluster outputted by Algorithm 3 that is a superset of C_i and does not contain any other 2-MPR cluster.* [7] *Furthermore, the overall clustering returned by Algorithm 3 is an $O(\log^* n)$-approximation.*

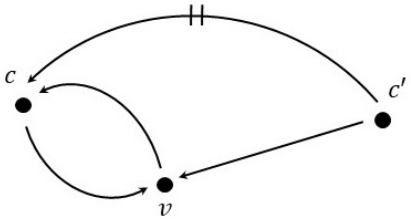
At the end of this section, we will also show an algorithm that outputs an optimal cluster C_i exactly, if C_i and any optimal cluster near C_i are 2-MPR.

Approximation algorithm for asymmetric k-center We start with a recap of the $O(\log^* n)$-approximation algorithm by Vishwanathan [1996]. This was the first nontrivial algorithm for asymmetric k-center, and the approximation ratio was later proven to be tight by Chuzhoy et al. [2005]. To explain the algorithm, it is convenient to think of asymmetric k-center as a set covering problem. Given an asymmetric k-center instance (S, d), define the directed graph $D_{(S,d)} = (S, A)$, where $A = \{(u, v) \mid d(u, v) \leq r^*\}$. For a point $v \in S$, we define $\Gamma_{\text{in}}(v)$ and $\Gamma_{\text{out}}(v)$ as the set of vertices with an arc to and from v, respectively. The asymmetric k-center problem is equivalent to finding a subset $C \subseteq S$ of size k such that $\cup_{c \in C} \Gamma_{\text{out}}(c) = S$. We also define $\Gamma^x_{\text{in}}(v)$ and $\Gamma^x_{\text{out}}(v)$ as the set of vertices which have a path of length $\leq x$ to and from v in $D_{(S,d)}$, respectively, and we define $\Gamma^x_{\text{out}}(A) = \bigcup_{v \in A} \Gamma^x_{\text{out}}(v)$ for a set $A \subseteq S$, and similarly for $\Gamma^x_{\text{in}}(A)$. It is standard to assume the value of r^* is known; since it is one of $O(n^2)$ distances, the algorithm can search for the correct value in polynomial time. Vishwanathan [1996] uses the following concept.

[7] Formally, given the output clustering $\mathcal{C}'$ of Algorithm 3, for all 2-MPR clusters C_i and C_j, there exists $C'_i \in \mathcal{C}'$ such that $C_i \subseteq C'_i$ and $C_j \cap C'_i = \emptyset$.



(a) u is a CCV, so it is distance $2r^*$ to its entire cluster.



(b) c satisfies CCV-proximity, so it is closer to v than c' is to v.

Figure 2.2: Examples of a center-capturing vertex (left), and CCV-proximity (right).

Definition 2.5.3. *Given an asymmetric k-center clustering instance (S, d), a point $v \in S$ is a* center-capturing vertex *(CCV) if $\Gamma_{in}(v) \subseteq \Gamma_{out}(v)$. In other words, for all $u \in S$, $d(u, v) \leq r^*$ implies $d(v, u) \leq r^*$.*

As the name suggests, each CCV $v \in C_i$, "captures" its center, i.e. $c_i \in \Gamma_{\text{out}}(v)$ (see Figure 2.2a). Therefore, v's entire cluster is contained inside $\Gamma^2_{\text{out}}(v)$, which is a nice property that the approximation algorithm exploits. At a high level, the approximation algorithm has two phases. In the first phase, the algorithm iteratively picks a CCV v arbitrarily and removes all points in $\Gamma^2_{\text{out}}(v)$. This continues until there are no more CCVs. For every CCV picked, the algorithm is guaranteed to remove an entire optimal cluster. In the second phase, the algorithm runs $\log^* n$ rounds of a greedy set-cover subroutine on the remaining points. See Algorithm 2. To prove the second phase terminates in $O(\log^* n)$ rounds, the analysis crucially assumes there are no CCVs among the remaining points. We refer the reader to [Vishwanathan, 1996] for these details.

Description of our algorithm and analysis We show a modification to the approximation algorithm of Vishwanathan [1996] leads to simultaneous guarantees in the worst case and under local perturbation resilience. Note that the set of all CCV's is identical to the symmetric set A defined in Section 2.3.2. In Section 2.3.2, we showed that all centers are in A, therefore, all centers are CCV's, assuming 2-PR. In this section, we have that each 2-MPR center is a CCV (Lemma 2.5.5), which is true by definition of r^*, ($C_i \subseteq \Gamma_{\text{out}}(c_i)$) and by using the definition of 2-MPR ($\Gamma_{\text{in}}(c_i) \subseteq C_i$). Since each 2-MPR center is a CCV, we might hope that we can output the 2-MPR clusters by iteratively choosing a CCV v and removing all points in $\Gamma^2_{\text{out}}(v)$. However, using this approach we might remove two or more 2-MPR centers in the same iteration, which means we would not output one separate cluster for each 2-MPR cluster. If we try to get around this problem by iteratively choosing a CCV v and removing all points in $\Gamma^1_{\text{out}}(v)$, then we may not remove one full cluster in each iteration, so for example, some of the 2-SPR clusters may be cut in half.

The key challenge is thus carefully specifying which nearby points get marked by each CCV c chosen by the algorithm. We fix this problem with two modifications that carefully balance the two guarantees. First, any CCV c chosen will mark points in the following way: for all $c' \in \Gamma_{\text{in}}(c)$, mark all points in $\Gamma_{\text{out}}(c')$. Intuitively, we still mark points that are two hops from c, but the first

Algorithm 2 $O(\log^* n)$ APPROXIMATION ALGORITHM FOR ASYMMETRIC k-CENTER [VISHWANATHAN, 1996]

Input: Asymmetric k-center instance (S, d), optimal radius r^* (or try all possible candidates)

Set $C = \emptyset$

Phase I: Pull out arbitrary CCVs

While there exists an unmarked CCV

- Pick an unmarked CCV c, add c to C, and mark all vertices in $\Gamma^2_{\text{out}}(c)$

Phase II: Recursive set cover

Set $A_0 = S \setminus \Gamma^5_{\text{out}}(C)$, $i = 0$.

While $|A_i| > k$:

- Set $A'_{i+1} = \emptyset$.
- While there exists an unmarked point in A_i:
  - Pick $v \in S$ which maximizes $\Gamma^5_{\text{out}}(v) \cap A_i$, mark points in $\Gamma^5_{\text{out}}(v) \cap A_i$, and add v to A'_{i+1}.
- Set $A_{i+1} = A'_{i+1} \cap A_0$ and $i = i + 1$

Output: Centers $C \cup A_{i+1}$

'hop' must go backwards, i.e., mark v such that there exists c' and $d(c', c) \leq r^*$ and $d(c', v) \leq r^*$. This gives us a useful property: if the algorithm picks a CCV $c \in C_i$ and it marks a different 2-MPR center c_j, then the middle hop must be a point q in C_j. However, we know from perturbation resilience that $d(c_j, q) < d(c, q)$. This fact motivates the final modification to the algorithm. Instead of picking arbitrary CCVs, we require the algorithm to choose CCVs with an extra structural property which we call *CCV-proximity* (Definition 2.5.4). See Figure 2.2b. Intuitively, a point c satisfying CCV-proximity must be closer than other CCVs to each point in $\Gamma_{\text{in}}(c)$. Going back to our previous example, c will NOT satisfy CCV-proximity because c_j is closer to q, but we will be able to show that all 2-MPR centers do satisfy CCV-proximity. Thus Algorithm 3 works as follows. It first chooses points satisfying CCV-proximity and marks points according to the rule mentioned earlier. When there are no more points satisfying CCV-proximity, the algorithm chooses regular CCVs. Finally, it runs Phase II as in Algorithm 2. This ensures that Algorithm 3 will output each 2-MPR center in its own cluster.

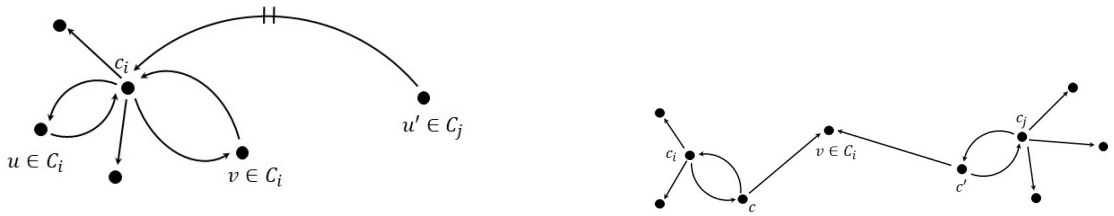
Details for Theorem 2.5.2 Now we formally define CCV-proximity. The other properties in the following definition, *center-separation* and *cluster-proximity*, are defined in terms of the optimal clustering, so they cannot be explicitly used by an algorithm, but they will simplify all of our proofs.

Definition 2.5.4. *1. An optimal center c_i satisfies* center-separation *if any point within distance r^* of c_i belongs to its cluster C_i. That is, $\Gamma_{in}(c_i) \subseteq C_i$ (see Figure 2.3a).* [8]

2. A CCV $c \in C_i$ satisfies CCV-proximity *if, given a CCV $c' \notin C_i$ and a point $v \in C_i$, we have $\alpha d(c, v) < d(c', v)$ (see Figure 2.3b).* [9]

[8] Center-separation is the local-PR equivalent of property 2 from Section 2.3.2.

[9] This is a generalization of α-center proximity [Awasthi et al., 2012], a property defined over an entire clustering



(a) c_i satisfies center-separation, so $\Gamma_{\text{in}}(c_i) \subseteq C_i$.

(b) $c \in C_i$ satisfies cluster-proximity, so it is closer to $v \in C_i$ than $c' \in C_j$ is to v.

Figure 2.3: Examples of center-separation (left), and cluster-proximity (right).

3. *A CCV $c \in C_i$ satisfies* weak CCV-proximity *if, given a CCV $c' \notin C_i$ and a point $v \in C_i$, we have $d(c, v) < d(c', v)$ (see Figure 2.3b).*

4. *A point c satisfies* closure *if it is a CCV, and each point in $\Gamma_{in}(c)$ is closer to c than any CCV outside of $\Gamma_{out}(c)$. That is, for all points $v \in \Gamma_{in}(c)$ and CCVs $c' \notin \Gamma_{out}(c)$, $d(c, v) < d(c', v)$ (see Figure 2.2b)* [10]

Next we prove that all 2-MPR centers satisfy CCV-proximity and center-separation. We also prove a third condition which states a CCV in some cluster C_i is closer than a CCV outside of C_i to all points in C_i. This property will help us prove Theorem 2.5.2.

Lemma 2.5.5. *Given an asymmetric k-center clustering instance (S, d) and a 2-MPR cluster C_i,*
(1) *c_i satisfies center-separation,*
(2) *any CCV $c \in C_i$ satisfies weak CCV-proximity,*
(3) *c_i satisfies closure.*
Furthermore, an α-PR cluster satisfies α-CCV proximity.

Proof. Given an instance (S, d) and a 2-MPR cluster C_i, we show that C_i has the desired properties.

Center separation: Assume there exists a point $v \in C_j$ for $j \neq i$ such that $d(v, c_i) \leq r^*$. The idea is to construct a 2-perturbation in which v becomes the center for C_i.

$$d''(s,t) = \begin{cases} \min(2r^*, 2d(s,t)) & \text{if } s = v, t \in C_i \\ 2d(s,t) & \text{otherwise.} \end{cases}$$

d'' is a valid 2-perturbation of d because for each point $u \in C_i$, $d(v, u) \leq d(v, c_i) + d(c_i, u) \leq 2r^*$. Define d' as the metric completion of d''. Then by Lemma 2.4.1, d' is a 2-metric perturbation with optimal cost $2r^*$. The set of centers $\{c_{i'}\}_{i'=1}^{k} \setminus \{c_i\} \cup \{v\}$ achieves the optimal cost, since v is distance $2r^*$ from C_i, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $2r^*$). If v is a noncenter, then $\{c_{i'}\}_{i'=1}^{k} \setminus \{c_i\} \cup \{v\}$ is a valid set of k centers. If $v = c_j$, then add an arbitrary point $v' \in C_j$ to this set of centers (it still achieves the optimal cost since adding another

instance, which states for all i, for all $v \in C_i$, $j \neq i$, we have $\alpha d(c_i, v) < d(c_j, v)$. We generalize to local-PR, asymmetric instances, and general CCV's.

[10] This is similar to a property in [Balcan and Liang, 2016].

center can only decrease the cost). Then in this new optimal clustering, c_i's center is a point in $\{c_{i'}\}_{i'=1}^{k} \setminus \{c_i\} \cup \{v, v'\}$, none of which are from C_i. We conclude that C_i is no longer an optimal cluster, contradicting 2-MPR.

Cluster-proximity: Given a CCV $c \in C_i$, a CCV $c' \in C_j$ such that $j \neq i$, and a point $v \in C_i$, assume to the contrary that $d(c', v) \leq d(c, v)$. We will construct a perturbation in which c and c' become centers of their respective clusters, and then v switches clusters. Define the following perturbation d''.

$$d''(s,t) = \begin{cases} \min(2r^*, 2d(s,t)) & \text{if } s = c, t \in C_i \text{ or } s = c', t \in C_j \cup \{v\} \\ 2d(s,t) & \text{otherwise.} \end{cases}$$

d'' is a valid 2-perturbation of d because for each point $u \in C_i$, $d(c,u) \leq d(c,c_i) + d(c_i,u) \leq 2r^*$, for each point $u \in C_j$, $d(c',u) \leq d(c',c_j) + d(c_j,u) \leq 2r^*$, and $d(c',v) \leq d(c,v) \leq d(c,c_i) + d(c_i,v) \leq 2r^*$. Define d' as the metric completion of d''. Then by Lemma 2.4.1, d' is a 2-metric perturbation with optimal cost $2r^*$. The set of centers $\{c_{i'}\}_{i'=1}^{k} \setminus \{c_i, c_j\} \cup \{c, c'\}$ achieves the optimal cost, since c and c' are distance $2r^*$ from C_i and C_j, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $2r^*$). Then since $d'(c', v) \leq d(c, v)$, v can switch clusters, contradicting perturbation resilience.

CCV-proximity: First we show that c_i is a CCV. By center-separation, we have that $\Gamma_{\text{in}}(c_i) \subseteq C_i$, and by definition of r^*, we have that $C_i \subseteq \Gamma_{\text{out}}(c_i)$. Therefore, $\Gamma_{\text{in}}(c_i) \subseteq C_i \subseteq \Gamma_{\text{out}}(c_i)$, so c_i is a CCV. Now given a point $v \in \Gamma_{\text{in}}(c_i)$ and a CCV $c \notin \Gamma_{\text{out}}(c_i)$, from center-separation and definition of r^*, we have $v \in C_i$ and $c \in C_j$ for $j \neq i$. Then from cluster-proximity, $d(c_i, v) < d(c, v)$. □

Algorithm 3 ALGORITHM FOR ASYMMETRIC k-CENTER UNDER PERTURBATION RESILIENCE

Input: Asymmetric k-center instance (S, d), distance r^* (or try all possible candidates)

Set $C = \emptyset$.

Phase I: Pull out special CCVs

- While there exists an unmarked CCV:
  - Pick an unmarked point c which satisfies CCV-proximity. If no such c exists, then pick an arbitrary unmarked CCV instead. Add c to C, and $\forall c' \in \Gamma_{\text{in}}(c)$, mark $\Gamma_{\text{out}}(c')$.
- For each $c \in C$, let V_c denote c's Voronoi tile of the marked points induced by C.

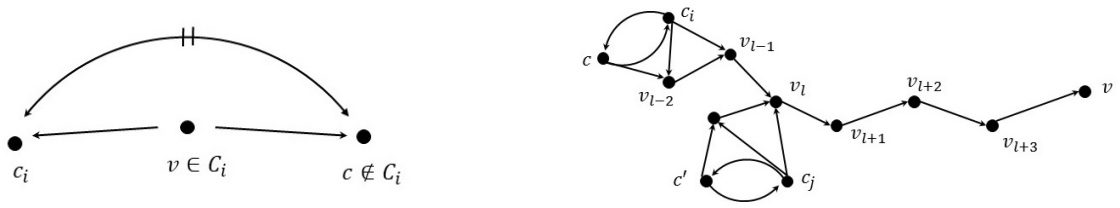
Phase II: Recursive set cover

- Run Phase II as in Algorithm 2, outputting A_{i+1}.
- Compute the Voronoi diagram $\{V'_c\}_{c \in C \cup A_{i+1}}$ of $S \setminus \Gamma^5_{\text{out}}(C)$ induced by $C \cup A_{i+1}$
- For each c in C, set $V'_c = V_c \cup V'_c$

Output: Sets $\{V'_c\}_{c \in C \cup A_{i+1}}$

Now using Lemma 2.5.5, we can prove Theorem 2.5.2.

Proof of Theorem 2.5.2. First we explain why Algorithm 3 retains the approximation guarantee of Algorithm 2. Given any CCV $c \in C_i$ chosen in Phase I, since c is a CCV, then $c_i \in \Gamma_{\text{out}}(c)$, and by



(a) A point $c \notin C_i$ cannot mark c_i without causing a contradiciton.

(b) Case 2 in the proof of Theorem 2.5.7: c' is closer to v than c is to v.

Figure 2.4: Example of Algorithm 3 (left), and the proof of Theorem 2.5.7 (right).

definition of r^*, $C_i \subseteq \Gamma_{\text{out}}(c_i)$. Therefore, each chosen CCV always marks its cluster, and we start Phase II with no remaining CCVs. This condition is sufficient for Phase II to return an $O(\log^* n)$ approximation (Theorem 3.1 from [Vishwanathan, 1996]).

Next we claim that for each 2-MPR cluster C_i, there exists a cluster outputted by Algorithm 3 that is a superset of C_i and does not contain any other 2-MPR cluster. To prove this claim, we first show there exists a point from C_i satisfying CCV-proximity that cannot be marked by any point from a different cluster in Phase I. From Lemma 2.5.5, c_i satisfies CCV-proximity and center-separation. If a point $c \notin C_i$ marks c_i, then $\exists v \in \Gamma_{\text{in}}(c) \cap \Gamma_{\text{in}}(c_i)$. By center-separation, $c_i \notin \Gamma_{\text{out}}(c)$, and therefore since c is a CCV, $c \notin \Gamma_{\text{out}}(c_i)$. But then from the definition of CCV-proximity for c_i and c, we have $d(c, v) < d(c_i, v)$ and $d(c_i, v) < d(c, v)$, so we have reached a contradiction (see Figure 2.4a).

At this point, we know a point $c \in C_i$ will always be chosen by the algorithm in Phase I. To finish the proof, we show that each point v from C_i is closer to c than to any other point $c' \notin C_i$ chosen as a center in Phase I. Since c and c' are both CCVs, this follows directly from cluster-proximity. [11] □

Strong perturbation resilience

Theorem 2.5.2 shows that Algorithm 3 will output each 2-PR center in its own cluster. Given some 2-PR center c_i, it is unavoidable that c_i might mark a *non* 2-PR center c_j, and capture all points in its cluster. Now we define a natural strengthening of α-PR which allows us to prove a stronger result than Theorem 2.5.2. Intuitively, an optimal cluster C_i satisfies *α-strong perturbation resillience* if all nearby clusters satisfy α-perturbation resilience. We show that Algorithm 3 with a slight modification outputs each 2-strong perturbation resilient cluster exactly.

Definition 2.5.6. *(Strong perturbation resilience) Given a k-center clustering instance (S, d), a subset C satisfies* α-strong perturbation resilience (α-SPR) *if there exists $S' \supseteq C$ which is α-PR, and furthermore, for each $u \in C$, if u' is in the same optimal cluster as u and if $d(u', v) \leq r^*$ then $v \in S'$.*

Intuitively, the nearby 2-PR clusters 'shield' C_i from all other points (see Figure 2.5).

[11] It is possible that a center c' chosen in Phase 2 may be closer to v than c is to v, causing c' to "steal" v; this is unavoidable. This is why Algorithm 3 separately computes the voronoi tiling from Phase I and Phase II, and so the final output is technically not a valid voronoi tiling over the entire instance S.

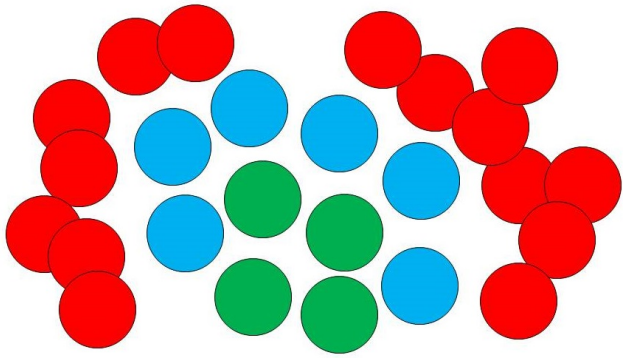


Figure 2.5: The red clusters are optimal clusters with no structure, the blue clusters are 2-PR clusters, and the green clusters are 2-PR clusters who only have neighbors which are also 2-PR (Theorem 2.5.7). Algorithm 4 outputs the green clusters exactly.

The only modification is that at the end of Phase II, instead of calculating the Voronoi diagram using the metric d, we assign each point $v \in S \setminus \Gamma^5_{\text{out}}(C)$ to the point in $C \cup A_{i+1}$ which minimizes the path length in $D_{(S,d)}$, breaking ties by distance to first common vertex in the shortest path.

Theorem 2.5.7. *Given an asymmetric k-center clustering instance (S, d) with optimal clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, consider a 2-PR cluster C_i. For all C_j, if there exists $u \in C_i$ and $v \in C_j$ such that $d(u, v) \leq r^*$, then C_j is 2-PR, then Algorithm 4 returns C_i exactly (if all neighboring optimal clusters of C_i are 2-PR, then C_i is included in the output of Algorithm 4).*

Proof. Given a 2-PR cluster C_i with the property in the theorem statement, by Theorem 2.5.2, there exists a CCV $c \in C_i$ from Phase I satisfying CCV-proximity such that $C_i \subseteq V_c$. Our goal is to show that $V_c = C_i$. First we show that $\Gamma_{\text{in}}(c) \subseteq C_i$, which will help us prove the theorem. Assume towards contradiction that there exists a point $v \in \Gamma_{\text{in}}(c) \setminus C_i$. Let $v \in C_j$. Since c is a CCV, we have $v \in \Gamma_{\text{out}}(c)$, so C_j must be 2-PR by definition. By Properties *(2)* and *(3)* of Lemma 2.5.5, c_j is a CCV and $d(c_j, v) < d(c, v)$. But this violates CCV-proximity of c, so we have reached a contradiction. Therefore, $\Gamma_{\text{in}}(c) \subseteq C_i$.

To finish the proof, we must show that $V_c \subseteq C_i$. Assume towards contradiction there exists $v \in V_c \setminus C_i$ at the end of the algorithm.

Case 1: v was marked by c in Phase I. Let $v \in C_j$. Then there exists a point $u \in \Gamma_{\text{in}}(c)$ such that $v \in \Gamma_{\text{out}}(u)$. From the previous paragraph we have that $\Gamma_{\text{in}}(c) \subseteq C_i$, so $u \in C_i$. Therefore, $v \in \Gamma_{\text{out}}(u)$ implies C_j must be 2-PR. Since v is from a different 2-PR cluster, it cannot be contained in V_c, so we have reached a contradiction.

Case 2: v was not marked by c in Phase I. Denote the shortest path in $D_{(S,d)}$ from c to v by $(c = v_0, v_1, \ldots, v_{L-1}, v_L = v)$. Let $v_\ell \in C_j$ denote the first vertex on the shortest path that is not in C_i (such a vertex must exist because $v \notin C_i$). Then $v_{\ell-1} \in C_i$ and $d(v_{\ell-1}, v_\ell) \leq r^*$, so C_j is 2-PR by the assumption in Theorem 2.5.7. Let c' denote the CCV chosen in Phase I such that $C_j \subseteq V_{c'}$. Then by Property *(2)* of Lemma 2.5.5, we have $d(c', v_\ell) < d(c, v_\ell)$.

Case 2a: $d(c, v_\ell) \leq r^*$. Then v_ℓ is the first vertex on the shortest path from c to v and c' to v, so v_ℓ is the first common vertex on the shortest paths. Since $d(c', v_\ell) < d(c, v_\ell)$, the algorithm will choose c' as the center for v. See Figure 2.4b.

Algorithm 4 OUTPUTTING OPTIMAL CLUSTERS FOR ASYMMETRIC k-CENTER UNDER STABILITY

Input: Asymmetric k-center instance (S, d), distance r^* (or try all possible candidates)

Set $C = \emptyset$.

Phase I: Pull out special CCVs

- While there exists an unmarked CCV:
  - Pick an unmarked point c which satisfies CCV-proximity. If no such c exists, then pick an arbitrary unmarked CCV instead. Add c to C, and $\forall c' \in \Gamma_{\text{in}}(c)$, mark $\Gamma_{\text{out}}(c')$.
- For each $c \in C$, let V_c denote c's Voronoi tile of the marked points induced by C.

Phase II: Recursive set cover

- Run Phase II as in Algorithm 2, outputting A_{i+1}.

Phase III: Assign points to centers

- For each $v \in S \setminus \Gamma^5_{\text{out}}(C)$, assign v to the center $c \in C \cup A_{i+1}$ with the minimum path length in $D_{(S,d)}$ from c to v, breaking ties by distance to first common vertex in the shortest path.
- Let V'_c denote the set of vertices in $v \in S \setminus \Gamma^5_{\text{out}}(C)$ assigned to c.
- For each c in C, set $V'_c = V_c \cup V'_c$

Output: Sets $\{V'_c\}_{c \in C \cup A_{i+1}}$

Case 2b: $d(c, v_\ell) > 2r^*$. But since $c' \in C_j$ is a CCV, we have $d(c', v_\ell) \leq d(c', c_j) + d(c_j, v_\ell) \leq 2r^*$, so the shortest path from c' to v_ℓ is at most 2, and the shortest path from c to v_ℓ is at least 3. Since v_ℓ is on the shortest path from c to v, it follows that the shortest path from c' to v is strictly shorter than the shortest path from c to v.

Case 2c: $r^* < d(c, v_\ell) \leq 2r^*$. In this case, we will show that $d(c', v_\ell) \leq r^*$, and therefore we conclude the shortest path from c' to v is strictly shorter than the shortest path from c to v, as in Case 2b. Assume towards contradiction that $d(c', v_\ell) > r^*$. Then we will create a 2-perturbation in which c and c' become centers for their own clusters, and v_ℓ switches clusters. Define the following perturbation d'.

$$d'(s,t) = \begin{cases} \min(2r^*, 2d(s,t)) & \text{if } s = c,\, t \in C_i \text{ or } s = c',\, t \in C_j \setminus \{v_\ell\} \\ d(s,t) & \text{if } s = c,\, t = v_\ell \\ 2d(s,t) & \text{otherwise.} \end{cases}$$

d' is a valid 2-perturbation of d because for each point $u \in C_i$, $d(c, u) \leq d(c, c_i) + d(c_i, u) \leq 2r^*$, for each point $u \in C_j$, $d(c', u) \leq d(c', c_j) + d(c_j, u) \leq 2r^*$, and $d(c, v_\ell) \leq 2r^*$. Therefore, d' does not decrease any distances (and by construction, d' does not increase any distance by more than a factor of 2). If the optimal cost is $2r^*$, then the set of centers $\{c_{i'}\}_{i'=1}^k \setminus \{c_i, c_j\} \cup \{c, c'\}$ achieves the optimal cost, since c and c' are distance $2r^*$ from $C_i \cup \{v_\ell\}$ and C_j, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $2r^*$). Then by perturbation resilience, it must be the case that $d'(c', v_\ell) < d'(c, v_\ell)$, which implies $2d(c', v_\ell) < d(c, v_\ell)$. But $d(c', v_\ell) > r^*$ and

$d(c, v_\ell) \leq 2r^*$, so we have a contradiction. Now we assume the optimal cost of d' is less than $2r^*$. Note that all distances $d(s,t)$ were increased to $2d(s,t)$ or $\min(2d(s,t), 2r^*)$ except for $d(c, v_\ell)$. Therefore, c must be a center for v_ℓ under d', or else the optimal could would be exactly $2r^*$ by Lemma 2.4.1. But it contradicts perturbation resilience to have c and c_ℓ in the same optimal cluster under a 2-perturbation. This completes the proof. □

2.6 k-center under (α, ϵ)-perturbation resilience

In this section, we consider (α, ϵ)-perturbation resilience. First, we show that any 2-approximation algorithm for symmetric k-center must be optimal under $(3, \epsilon)$-perturbation resilience (Theorem 2.6.1). Next, we show how to extend this result to local perturbation resilience (Theorem 2.6.8). Then we give an algorithm for asymmetric k-center which returns a clustering that is ϵ-close to $\mathcal{OPT}$ under $(3, \epsilon)$-perturbation resilience (Theorem 2.6.17). For all of these results, we assume a lower bound on the size of the optimal clusters, $|C_i| > 2\epsilon n$ for all $i \in [k]$. We show the lower bound on cluster sizes is necessary; in its absence, the problem becomes NP-hard for all values of $\alpha \geq 1$ and $\epsilon > 0$ (Theorem 2.6.18). The theorems in this section require a careful reasoning about sets of centers under different perturbations that cannot all simultaneously be optimal.

2.6.1 Symmetric k-center

We show that for any $(3, \epsilon)$-perturbation resilient k-center instance such that $|C_i| > 2\epsilon n$ for all $i \in [k]$, there cannot be any pair of points from different clusters which are distance $\leq r^*$. This structural result implies that simple algorithms will return the optimal clustering, such as running any 2-approximation algorithm or running the Single Linkage algorithm, which is a fast algorithm widely used in practice for its simplicity.

Theorem 2.6.1. *Given a $(3, \epsilon)$-perturbation resilient symmetric k-center instance (S, d) where all optimal clusters are size $> \max(2\epsilon n, 3)$, then the optimal clusters in $\mathcal{OPT}$ are exactly the connected components of the threshold graph $G_{r^*} = (S, E)$, where $E = \{(u, v) \mid d(u, v) \leq r^*\}$.*

First we explain the high-level idea behind the proof.

Proof idea. Since each optimal cluster center is distance r^* from all points in its cluster, it suffices to show that any two points in different clusters are greater than r^* apart from each other. Assume on the contrary that there exist $p \in C_i$ and $q \in C_{j \neq i}$ such that $d(p, q) \leq r^*$. First we find a set of $k + 2$ points and a 3-perturbation d', such that every size k subset of the points are optimal centers under d'. Then we show how this leads to a contradiction under $(3, \epsilon)$-perturbation resilience.

Here is how we find a set of $k + 2$ points and a perturbation d' such that all size k subsets are optimal centers under d'. From our assumption, p is distance $\leq 3r^*$ from every point in $C_i \cup C_j$ (by the triangle inequality). Under a 3-perturbation in which all distances are blown up by a factor of 3 except $d(p, C_i \cup C_j)$, then replacing c_i and c_j with p would still give us a set of $k - 1$ centers that achieve the optimal cost. But, *would this contradict $(3, \epsilon)$-perturbation resilience?* Indeed, not! Perturbation resilience requires exactly k *distinct* centers. [12] The key challenge is to pick a final

[12] This distinction is well-motivated; if for some application, the best k-center solution is to put two centers at the same location, then we could achieve the exact same solution with $k - 1$ centers. That implies we should have been running k'-center for $k' = k - 1$ instead of k.

"dummy" center to guarantee that the Voronoi partition is ϵ-far from $\mathcal{OPT}$. The dummy center might "accidentally" be the closest center for almost all points in C_i or C_j. Even worse, it might be the case that the new center sets off a chain reaction in which it becomes center to a cluster C_x, and c_x becomes center to C_j, which would also result in a partition that is not ϵ-far from $\mathcal{OPT}$.

To deal with the chain reactions, we crucially introduce the notion of a *cluster capturing center* (CCC). A cluster capturing center (CCC) is not to be confused with a center-capturing vertex (CCV), defined by Vishwanathan [1996] and used in the previous section. c_x is a CCC for C_y, if for all but ϵn points $p \in C_y$, $d(c_x, p) \leq r^*$ and for all $i \neq x, y$, $d(c_x, p) < d(c_i, p)$. Intuitively, a CCC exists if and only if c_x is a valid center for C_y when c_y is taken out of the set of optimal centers (i.e., a chain reaction will occur). We argue that if a CCC does not exist then every dummy center we pick must be close to either C_i or C_j, since there are no chain reactions. If there does exist a CCC c_x for C_y, then it is much harder to reason about what happens to the dummy centers under d', since there may be chain reactions. However, we can define a new d'' by increasing all distances except $d(c_x, C_y)$, which allows us to take c_y out of the set of optimal centers, and then any dummy center must be close to C_x or C_y. There are no chain reactions because we already know c_x is the best center for C_y among the original optimal centers. Thus, whether or not there exists a CCC, we can find $k + 2$ points close to the entire dataset by picking points from both C_i and C_j (resp. C_x and C_y).

Because of the assumption that all clusters are size $> 2\epsilon n$, for every 3-perturbation there must be a bijection between clusters and centers, where the center is closest to the majority of points in the corresponding cluster. We show that all size k subsets of the $k+2$ points cannot simultaneously admit bijections that are consistent with one another.

Formal analysis. We start out with a simple implication from the assumption that $|C_i| > 2\epsilon n$ for all i.

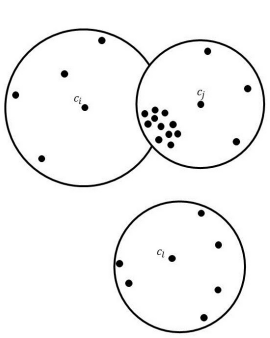
Fact 2.6.2. *Given a clustering instance which is (α, ϵ)-perturbation resilient for $\alpha \geq 1$, and all optimal clusters have size $> 2\epsilon n$, then for any α-perturbation d', for any set of optimal centers $c'_1, \ldots, c'_k$ of d', for each optimal cluster C_i, there must be a unique center c'_i which is the center for more than half of the points in C_i under d'.*

This fact follows simply from the definition of (α, ϵ)-perturbation resilience (under d', at most ϵn points in the optimal solution can change clusters), and the assumption that all optimal clusters are size $> 2\epsilon n$. Now we formally define a CCC.

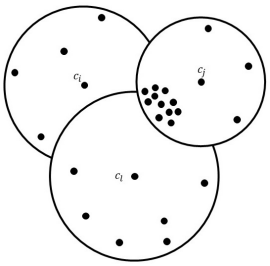
Definition 2.6.3. *A center c_i is a first-order cluster-capturing center (CCC) for C_j if for all $x \neq j$, for more than half of the points $p \in C_j$, $d(c_i, p) < d(c_x, p)$ and $d(c_i, p) \leq r^*$ (see Figure 2.6a). c_i is a second-order cluster-capturing center (CCC2) for C_j if there exists a c_l such that for all $x \neq j, l$, for more than half of points $p \in C_j$, $d(c_i, p) < d(c_x, p)$ and $d(c_i, p) \leq r^*$ (see Figure 2.6b).*

Each cluster C_j can have at most one CCC c_i because c_i is closer than any other center to more than half of C_j. Every CCC is a CCC2, since the former is a stronger condition. However, it is possible for a cluster to have multiple CCC2's. [13] We needed to define CCC2 for the following reason. Assuming there exist $p \in C_i$ and $q \in C_j$ which are close, and we replace c_i and c_j with p in

[13] In fact, a cluster can have at most three CCC2's, but we do not use this in our analysis.



(a) c_i *is a CCC for* C_j.



(b) c_i *is a CCC for* C_j *and* c_x *is a CCC2 for* C_j.

Figure 2.6: (a) Definition of a CCC, and (b) definition of a CCC2.

the set of centers. It is possible that c_j is a CCC for C_i, but this does not help us, since we want to analyze the set of centers after removing c_j. However, if we know that c_x is a CCC2 (it is the best center for C_i, disregarding c_j), then we know that c_x will be the best center for C_i after replacing c_i and c_j with p. Now we use this definition to show that if two points from different clusters are close, we can find a set of $k+2$ points and a 3-perturbation d', such that every size k subset of the points are optimal centers under d'. To formalize this notion, we give one more definition.

Definition 2.6.4. *A set $C \subseteq S$ (β, γ)-hits S if for all $s \in S$, there exist β points in C at distance $\leq \gamma r^*$ to s.*

Note that if a set C of $k+2$ points $(3,3)$-hits S, then any size k subset of C is still $3r^*$ from every point in S, and later we will show that means there exists a perturbation d' such that every size k subset must be an optimal set of centers.

Lemma 2.6.5. *Given a clustering instance satisfying $(3, \epsilon)$-perturbation resilience such that all optimal clusters are size $> 2\epsilon n$ and there are two points from different clusters which are $\leq r^*$ apart from each other, then there exists a set $C \subseteq S$ of size $k+2$ which $(3,3)$-hits S.*

Proof. First we prove the lemma assuming that a CCC2 exists, and then we prove the other case. When a CCC2 exists, we do not need the assumption that two points from different clusters are close.

Case 1: There exists a CCC2. If there exists a CCC, then denote c_x as a CCC for C_y. If there does not exist a CCC, then denote c_x as a CCC2 for C_y. We will show that all points are close to either C_x or C_y. c_x is distance $\leq r^*$ to all but ϵn points in C_y. Therefore, $d(c_x, c_y) \leq 2r^*$ and so c_x is distance $\leq 3r^*$ to all points in C_y. Consider the following d'.

$$d'(s,t) = \begin{cases} \min(3r^*, 3d(s,t)) & \text{if } s = c_x, t \in C_y \\ 3d(s,t) & \text{otherwise.} \end{cases}$$

This is a 3-perturbation because $d(c_x, C_y) \leq 3r^*$. Then by Lemma 2.2.8, the optimal cost is $3r^*$. Given any $p \in S$, the set of centers $\{c_l\}_{i=1}^k \setminus \{c_y\} \cup \{p\}$ achieves the optimal cost, since c_x is distance $3r^*$ from C_y, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius

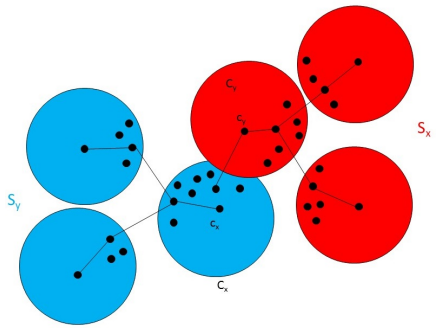


Figure 2.7: *Case 1 of Lemma 2.6.5*

$3r^*$). Therefore, this set of centers must create a partition that is ϵ-close to $\mathcal{OPT}$, or else there would be a contradiction. Then from Fact 2.6.2, one of the centers in $\{c_l\}_{i=1}^k \setminus \{c_y\} \cup \{p\}$ must be the center for the majority of points in C_y under d'. If this center is c_l, $l \neq x, y$, then for the majority of points $q \in C_y$, $d(c_l, q) \leq r^*$ and $d(c_l, q) < d(c_z, q)$ for all $z \neq l, y$. Then by definition, c_l is a CCC for C_y. But then l must equal x, so we have a contradiction. Note that if some c_l has for the majority of $q \in C_y$, $d(c_l, q) \leq d(c_z, q)$ (non-strict inequality) for all $z \neq l, y$, then there is another equally good partition in which c_l is not the center for the majority of points in C_y, so we still obtain a contradiction. Therefore, either p or c_x must be the center for the majority of points in C_y under d'.

If c_x is the center for the majority of points in C_y, then p must be the center for the majority of points in C_x (it cannot be a different center c_ℓ, since c_x is a better center for C_x than c_ℓ by definition). Therefore, each $p \in S$ is distance $\leq r^*$ to all but ϵn points in either C_x or C_y.

Now partition all the non-centers into two sets S_x and S_y, such that

$$S_x = \{p \mid \text{for the majority of points } q \in C_x,\ d(p, q) \leq r^*\}, \text{ and}$$
$$S_y = \{p \mid p \notin S_x \text{ and for the majority of points } q \in C_y,\ d(p, q) \leq r^*\}.$$

Then given $p, q \in S_x$, there exists an $s \in C_x$ such that $d(p, q) \leq d(p, s) + d(s, q) \leq 2r^*$ (since both points are close to more than half of points in C_x). Similarly, any two points $p, q \in S_y$ are $\leq 2r^*$ apart. See Figure 2.7.

Now we will find a set of $k + 2$ points that $(3, 3)$-hits S. For now, assume that S_x and S_y are both nonempty. Given an arbitrary pair $p \in S_x$, $q \in S_y$, we claim that $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S. Given a non-center $s \in C_i$ such that $i \neq x$ and $i \neq y$, without loss of generality let $s \in S_x$. Then c_i, p, and c_x are all distance $3r^*$ to s. Furthermore, c_i, p and c_x are all distance $3r^*$ to c_i. Given a point $s \in C_x$, then c_x, c_y, and p are distance $3r^*$ to s because $d(c_x, c_y) \leq 2r^*$, and a similar argument holds for $s \in C_y$. Therefore, $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S.

If $S_x = \emptyset$ or $S_y = \emptyset$, then we can prove a slightly stronger statement: for each pair of non-centers $\{p, q\}$, $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S. Without loss of generality, let $S_y = \emptyset$. Given a point $s \in C_i$ such that $i \neq x$ and $i \neq y$, then c_i, c_x, and p are all distance $3r^*$ to s. Given a point $s \in C_x$, then p, c_x, and c_y are all distance $\leq 3r^*$ to s. Given a point $s \in C_y$, then p, c_x, and c_y are all distance $\leq 3r^*$ to s because $s, p \in S_x$ implies $d(s, p) \leq 2r^*$. Thus, we have proven case 1.

Case 2: There does not exist a CCC2. Now we use the assumption that there exist $p \in C_x$,

$q \in C_y$, $x \neq y$, such that $d(p,q) \leq r^*$. Then by the triangle inequality, p is distance $\leq 3r^*$ to all points in C_x and C_y. Consider the following d'.

$$d'(s,t) = \begin{cases} \min(3r^*, 3d(s,t)) & \text{if } s = p,\, t \in C_x \cup C_y \\ 3d(s,t) & \text{otherwise.} \end{cases}$$

This is a 3-perturbation because $d(p, C_x \cup C_y) \leq 3r^*$. Then by Lemma 2.2.8, the optimal cost is $3r^*$. Given any $s \in S$, the set of centers $\{c_l\}_{i=1}^k \setminus \{c_x, c_y\} \cup \{p, s\}$ achieves the optimal cost, since p is distance $3r^*$ from $C_x \cup C_y$, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $3r^*$). Therefore, this set of centers must create a partition that is ϵ-close to $\mathcal{OPT}$, or else there would be a contradiction. Then from Fact 2.6.2, one of the centers in $\{c_l\}_{i=1}^k \setminus \{c_x, c_y\} \cup \{p, s\}$ must be the center for the majority of points in C_x under d'.

If this center is c_ℓ for $\ell \neq x$ and $\ell \neq y$, then for the majority of points $t \in C_x$, $d(c_\ell, t) \leq r^*$ and $d(c_\ell, t) < d(c_z, t)$ for all $z \neq \ell, x, y$. Then by definition, c_ℓ is a CCC2 for C_x, and we have a contradiction.

Similar logic applies to the center for the majority of points in C_y. Therefore, p and s must be the centers for C_x and C_y. Since s was an arbitrary noncenter, all noncenters are distance $\leq r^*$ to all but ϵn points in either C_x or C_y.

Similar to Case 1, we now partition all the non-centers into two sets S_x and S_y, such that

$$S_x = \{u \mid \text{for the majority of points } v \in C_x,\ d(u,v) \leq r^*\} \text{ and}$$
$$S_y = \{u \mid u \notin S_x \text{ and for the majority of points } v \in C_y,\ d(u,v) \leq r^*\}.$$

As before, each pair of points in S_x are distance $\leq 2r^*$ apart, and similarly for S_y. It is no longer true that $d(c_x, c_y) \leq 2r^*$, however, we can prove that for both S_x and S_y, there exist points from two distinct clusters each. From the previous paragraph, given a non-center $s \in C_i$ for $i \neq x, y$, we know that p and s are centers for C_x and C_y. With an identical argument, given $t \in C_j$ for $j \neq x, y, i$, we can show that q and t are centers for C_x and C_y. It follows that S_x and S_y both contain points from at least two distinct clusters.

Now we finish the proof by showing that for each pair $u \in S_x$, $v \in S_y$, $\{c_\ell\}_{\ell=1}^k \cup \{u, v\}$ $(3,3)$-hits S. Given a non-center $s \in C_i$, without loss of generality $s \in S_x$, then there exists $j \neq i$ and $t \in C_j \cap S_x$. Then c_i, c_j, and u are $3r^*$ to s and c_i, c_x, and u are $3r^*$ to c_i. In the case where $i = x$, then c_i, c_j, and u are $3r^*$ to c_i. This concludes the proof. □

So far, we have shown that by just assuming two points from different clusters are close, we can find a set of $k+2$ points that $(3,3)$-hits S. Now we will show that such a set leads to a contradiction under $(3,\epsilon)$-perturbation resilience. Specifically, we will show there exists a perturbation d' such that any size k subset can be an optimal set of centers. But it is not possible that all $\binom{k+2}{k}$ of these sets of centers simultaneously create partitions that are ϵ-close to $\mathcal{OPT}$. First we state a lemma which proves there does exist a perturbation d' such that any size k subset is an optimal set of centers.

Lemma 2.6.6. *Given a k-center clustering instance (S,d), given $z \geq 0$, and given a set $C \subseteq S$ of size $k+z$ which $(z+1,\alpha)$-hits S, there exists an α-metric perturbation d' such that all size k subsets of C are optimal sets of centers under d'.*

Proof. Consider the following perturbation d''.

$$d''(s,t) = \begin{cases} \min(\alpha r^*, \alpha d(s,t)) & \text{if } s \in C \text{ and } d(s,t) \leq \alpha r^* \\ \alpha d(s,t) & \text{otherwise.} \end{cases}$$

By Lemma 2.4.1, the metric closure d' of d'' is an α-metric perturbation with optimal cost αr^*. Given any size k subset $C' \subseteq C$, then for all $v \in S$, there is still at least one $c \in C'$ such that $d(c,v) \leq \alpha r^*$, therefore by construction, $d'(c,v) \leq \alpha r^*$. It follows that C' is a set of optimal centers under d'. □

Next, we state a fact that helps clusters rank their best centers from the set of $k+2$ points. For each cluster C_i, we would like to have a ranking of all points such that for a given d' and set of k centers, the center for C_i is the highest point in the ranking. The following fact shows this ranking is well-defined.

Fact 2.6.7. *Given a k-center clustering instance (S,d) such that all optimal clusters have size $> 2\epsilon n$, an α-perturbation d' of d, and a cluster C_x, there exists a ranking $R_{x,d'}$ of S such that for any set of optimal centers C, the center for C_x is the highest-ranked point in $R_{x,d'}$.* [14]

Proof. Assume the fact is false. Then there exists a d', a cluster C_i, two points p and q, and two sets of k centers $p, q \in C$ and $p, q \in C'$ which achieve the optimal cost under d', but p is the center for C_i in C while q is the center for C_i in C'. Then p is closer than all other points in C to all but ϵn points in C_i. Similarly, q is closer than all other points in C' to all but ϵn points in C_i. Since $|C_i| > 2\epsilon n$, this causes a contradiction. □

We also define $R_{x,d',C} : C \to [n']$ as the ranking specific to C, where $|C| = n'$. Now we can prove Theorem 2.6.1.

Proof of Theorem 2.6.1. It suffices to prove that any two points from different clusters are at distance $> r^*$ from each other. Assume towards contradiction that this is not true. Then by Lemma 2.6.5, there exists a set C of size $k+2$ which $(3,3)$-hits S. From Lemma 2.6.6, there exists a 3-metric perturbation d' such that all size k subsets of C are optimal sets of centers under d'. Consider the ranking of each cluster for C over d' guaranteed from Fact 2.6.7. We will show this ranking leads to a contradiction.

Consider the set of all points ranked 1 or 2 by any cluster, formally, $\{p \in C \mid \exists i \text{ s.t. } R_{i,d',C} \leq 2\}$. This set is a subset of C, since we are only considering the rankings of points in C, so it is size $\leq k+2$. Note that a point cannot be ranked both 1 and 2 by a cluster. Then as long as $k > 2$, it follows by the Pigeonhole Principle that there exists a point $c \in C$ which is ranked in the top two by two different clusters. Formally, there exists x and y such that $x \neq y$, $R_{x,d',C}(c) \leq 2$, and $R_{y,d',C}(c) \leq 2$. Denote u and v such that $R_{x,d',C}(u) = 1$ and $R_{y,d',C}(v) = 1$. If u or v is equal to c, then redefine it to an arbitrary center in $C \setminus \{c,u,v\}$. Consider the set of centers $C' = C \setminus \{u,v\}$ which is optimal under d' by construction. But then from Fact 2.6.7, c is the center for the majority of points in both C_x and C_y, contradicting Fact 2.6.2. This completes the proof. □

[14] Formally, for each C_x, there exists a bijection $R_{x,d'} : S \to [n]$ such that for all sets of k centers C that achieve the optimal cost under d', we have $c = \text{argmin}_{c' \in C} R_{x,d'}(c')$ if and only if $\text{Vor}_C(c)$ is ϵ-close to C_x.

2.6.2 Local perturbation resilience

Now we extend the argument from the previous section to local perturbation resilience. First we state our main structural result, which is that any pair of points from different $(3, \epsilon)$-PR clusters must be distance $> r^*$ from each other. Then we will show how the structural result easily leads to an algorithm for $(3, \epsilon)$-SPR clusters.

Theorem 2.6.8. *Given a k-center clustering instance (S, d) with optimal radius r^* such that all optimal clusters are size $> 2\epsilon n$ and there are at least three $(3, \epsilon)$-PR clusters, then for each pair of $(3, \epsilon)$-PR clusters C_i and C_j, for all $u \in C_i$ and $v \in C_j$, we have $d(u, v) > r^*$.*

Before we prove this theorem, we show how it implies an algorithm to output the optimal $(3, \epsilon)$-SPR clusters exactly. Since the distance from each point to its closest center is $\leq r^*$, a corollary of Theorem 2.6.8 is that any 2-approximate solution must contain the optimal $(3, \epsilon)$-SPR clusters, as long as the 2-approximation satisfies two sensible conditions: *(1)* for every point v and its assigned center u (so we know $d(u, v) \leq 2r^*$), $\exists w$ 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. [15]

Theorem 2.6.9. *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 $(3, \epsilon)$-SPR clusters.*

Proof. Given such a clustering instance, then Theorem 2.6.8 ensures that there is no edge of length r^* between points from two different $(3, \epsilon)$-PR clusters. Given a $(3, \epsilon)$-SPR cluster C_i, it follows that there is no point $v \notin C_i$ such that $d(v, C_i) \leq r^*$. Therefore, given a 2-approximate solution $\mathcal{C}$ satisfying condition *(1)*, any $u \in C_i$ and $v \notin C_i$ cannot be in the same cluster. This is because in the graph of datapoints where edges signify a distance $\leq r^*$, C_i is an isolated component. Finally, by condition *(2)*, C_i must not be split into two clusters. Therefore, $C_i \in \mathcal{C}$. □

Proof idea for Theorem 2.6.8 The high level idea of this proof is similar to the proof of Theorem 2.6.1. In fact, the first half is very similar to Lemma 2.6.5: we show that if two points from different PR clusters are close together, then there must exist a set of $k + 2$ points C which $(3, 3)$-hits the entire point set. In the previous section, we arrived at a contradiction by showing that it is not possible that all $\binom{k+2}{k}$ subsets of C can be centers that are ϵ-close to $\mathcal{OPT}$. However, the weaker local PR assumption poses a new challenge.

As in the previous section, we will still argue that all size k subsets of C cannot stay consistent with the $(3, \epsilon)$-PR clusters using a ranking argument which maps optimal clusters to optimal centers, but our argument will be to establish conditional claims which narrow down the possible sets of ranking lists. For instance, assume there is a $(3, \epsilon)$-PR cluster C_i which ranks c_i first, and ranks c_j second. Then under subsets C' which do not contain c_i, c_j is the center for a cluster C'_i which

[15] 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.

is ϵ-close to C_i. Therefore, a different point in C' must be the center for the majority of points in C_j (and it cannot be a different center c_ℓ without causing a contradiction). This is the basis for Lemma 2.6.13, which is the main workhorse lemma in the proof of Theorem 2.6.8. By building up conditional statements, we are able to analyze every possibility of the ranking lists for the three $(3, \epsilon)$-PR clusters and show that all of them lead to contradictions, proving Theorem 2.6.8.

Formal analysis of Theorem 2.6.8 We start with a local perturbation resilience variant of Fact 2.6.2.

Fact 2.6.10. *Given a k-center clustering instance (S, d) such that all optimal clusters have size $> 2\epsilon n$, let d' denote an α-perturbation with optimal centers $C' = \{c'_1, \ldots, c'_k\}$. Let $\mathcal{C}'$ denote the set of (α, ϵ)-PR clusters. Then there exists a one-to-one function $f : \mathcal{C}' \to C'$ such that for all $C_i \in \mathcal{C}'$, $|Vor_{C,d'}(f(C_i)) \cap C_i| \geq |C_i| - \epsilon n$. That is, the optimal cluster in d' whose center is $f(C_i)$ contains all but ϵn of the points in C_i.*

In words, for any set of optimal centers under an α-perturbation, each PR cluster can be paired to a unique center. This follows simply because all optimal clusters are size $> 2\epsilon n$, yet under a perturbation, $< \epsilon n$ points can switch out of each PR cluster. Because of this fact, for a perturbation d' with set of optimal centers C and an (α, ϵ)-PR cluster C_x, we will say that c is the center for C_x under d' if c is the center for the majority of points in C_x. Now we are ready to prove the first half of Theorem 2.6.8, stated in the following lemma. The proof is similar to Lemma 2.6.5.

Lemma 2.6.11. *Given a k-center clustering instance (S, d) such that all optimal clusters are size $> 2\epsilon n$ and there exist two points at distance r^* from different $(3, \epsilon)$-PR clusters, then there exists a partition $S_x \cup S_y$ of the non-centers $S \setminus \{c_\ell\}_{\ell=1}^k$ such that for all pairs $p \in S_x$, $q \in S_y$, $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S.*

Proof. This proof is split into two main cases. The first case is the following: there exists a CCC2 for a $(3, \epsilon)$-PR cluster, discounting a $(3, \epsilon)$-PR cluster. In fact, in this case, we do not need the assumption that two points from different PR clusters are close. If there exists a CCC to a $(3, \epsilon)$-PR cluster, denote the CCC by c_x and the cluster by C_y. Otherwise, let c_x denote a CCC2 to a $(3, \epsilon)$-PR cluster C_y, discounting a $(3, \epsilon)$-PR center c_z. Then c_x is at distance $\leq r^*$ to all but ϵn points in C_y. Therefore, $d(c_x, c_y) \leq 2r^*$ and so c_x is at distance $\leq 3r^*$ to all points in C_y. Consider the following perturbation d''.

$$d''(s,t) = \begin{cases} \min(3r^*, 3d(s,t)) & \text{if } s = c_x,\, t \in C_y \\ 3d(s,t) & \text{otherwise.} \end{cases}$$

This is a 3-perturbation because for all $v \in C_y$, $d(c_x, v) \leq 3r^*$. Define d' as the metric completion of d''. Then by Lemma 2.4.1, d' is a 3-metric perturbation with optimal cost $3r^*$. Given any non-center $v \in S$, the set of centers $\{c_\ell\}_{\ell=1}^k \setminus \{c_y\} \cup \{v\}$ achieves the optimal score, since c_x is at distance $3r^*$ from C_y, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $3r^*$). Therefore, from Fact 2.6.10, one of the centers in $\{c_\ell\}_{\ell=1}^k \setminus \{c_y\} \cup \{v\}$ must be the center for the majority of points in C_y under d'. If this center is c_ℓ, $\ell \neq x, y$, then for the majority of points $u \in C_y$, $d(c_\ell, u) \leq r^*$ and $d(c_\ell, u) < d(c_z, u)$ for all $z \neq \ell, y$. Then by definition, c_ℓ is a CCC for the $(3, \epsilon)$-PR cluster, C_y. But then by construction, ℓ must equal x, so we have a

contradiction. Note that if some c_ℓ has for the majority of $u \in C_y$, $d(c_\ell, u) \leq d(c_z, u)$ (non-strict inequality) for all $z \neq \ell, y$, then there is another equally good partition in which c_ℓ is not the center for the majority of points in C_y, so we still obtain a contradiction. Therefore, either v or c_x must be the center for the majority of points in C_y under d'.

If c_x is the center for the majority of points in C_y, then because C_y is $(3, \epsilon)$-PR, the corresponding cluster must contain fewer than ϵn points from C_x. Furthermore, since for all $\ell \neq x$ and $u \in C_x$, $d(u, c_x) < d(u, c_\ell)$, it follows that v must be the center for the majority of points in C_x. Therefore, every non-center $v \in S$ is at distance $\leq r^*$ to the majority of points in either C_x or C_y.

Now partition all the non-centers into two sets S_x and S_y, such that

$$S_x = \{u \mid \text{for the majority of points } v \in C_x,\ d(u, v) \leq r^*\}, \text{ and}$$
$$S_y = \{u \mid u \notin S_x \text{ and for the majority of points } v \in C_y,\ d(u, v) \leq r^*\}.$$

Given $p, q \in S_x$, there exists an $s \in C_x$ such that $d(p, q) \leq d(p, s) + d(s, q) \leq 2r^*$ (since both points are close to more than half of points in C_x). Similarly, any two points $p, q \in S_y$ are $\leq 2r^*$ apart.

Now we will find a set of $k + 2$ points that $(3, 3)$-hits S. For now, assume that S_x and S_y are both nonempty. Given an arbitrary pair $p \in S_x$, $q \in S_y$, we claim that $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S. Given a non-center $s \in C_i$ such that $i \neq x$ and $i \neq y$, without loss of generality let $s \in S_x$. Then c_i, p, and c_x are all distance $3r^*$ to s. Furthermore, c_i, p and c_x are all distance $3r^*$ to c_i. Given a point $s \in C_x$, then c_x, c_y, and p are distance $3r^*$ to s because $d(c_x, c_y) \leq 2r^*$, and a similar argument holds for $s \in C_y$. Therefore, $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S.

If $S_x = \emptyset$ or $S_y = \emptyset$, then we can prove a slightly stronger statement: for each pair of non-centers $\{p, q\}$, $\{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S. Without loss of generality, let $S_y = \emptyset$. Given a point $s \in C_i$ such that $i \neq x$ and $i \neq y$, then c_i, c_x, and p are all distance $3r^*$ to s. Given a point $s \in C_x$, then p, c_x, and c_y are all distance $\leq 3r^*$ to s. Given a point $s \in C_y$, then p, c_x, and c_y are all distance $\leq 3r^*$ to s because $s, p \in S_x$ implies $d(s, p) \leq 2r^*$. Thus, we have proven case 1.

Now we turn to the other case. Assume there does not exist a CCC2 to a PR cluster, discounting a PR center. In this case, we need to use the assumption that there exist $(3, \epsilon)$-PR clusters C_x and C_y, and $p \in C_x$, $q \in C_y$ such that $d(p, q) \leq r^*$. Then by the triangle inequality, p is distance $\leq 3r^*$ to all points in C_x and C_y. Consider the following d''.

$$d''(s, t) = \begin{cases} \min(3r^*, 3d(s, t)) & \text{if } s = p,\ t \in C_x \cup C_y \\ 3d(s, t) & \text{otherwise.} \end{cases}$$

This is a 3-perturbation because $d(p, C_x \cup C_y) \leq 3r^*$. Define d' as the metric completion of d''. Then by Lemma 2.4.1, d' is a 3-metric perturbation with optimal cost $3r^*$. Given any non-center $s \in S$, the set of centers $\{c_\ell\}_{\ell=1}^k \setminus \{c_x, c_y\} \cup \{p, s\}$ achieves the optimal score, since p is distance $3r^*$ from $C_x \cup C_y$, and all other clusters have the same center as in $\mathcal{OPT}$ (achieving radius $3r^*$).

From Fact 2.6.10, one of the centers in $\{c_\ell\}_{\ell=1}^k \setminus \{c_x, c_y\} \cup \{p, s\}$ must be the center for the majority of points in C_x under d'. If this center is c_ℓ for $\ell \neq x, y$, then for the majority of points $t \in C_x$, $d(c_\ell, t) \leq r^*$ and $d(c_\ell, t) < d(c_z, t)$ for all $z \neq \ell, x, y$. So by definition, c_ℓ is a CCC2 for C_x discounting c_y, which contradicts our assumption. Similar logic applies to the center for the

majority of points in C_y. Therefore, p and s must be the centers for C_x and C_y. Since s was an arbitrary non-center, all non-centers are distance $\leq r^*$ to all but ϵn points in either C_x or C_y.

Similar to Case 1, we now partition all the non-centers into two sets S_x and S_y, such that

$$S_x = \{u \mid \text{for the majority of points } v \in C_x,\ d(u,v) \leq r^*\} \text{ and}$$
$$S_y = \{u \mid u \notin S_x \text{ and for the majority of points } v \in C_y,\ d(u,v) \leq r^*\}.$$

As before, each pair of points in S_x are distance $\leq 2r^*$ apart, and similarly for S_y. It is no longer true that $d(c_x, c_y) \leq 2r^*$, however, we can prove that for both S_x and S_y, there exist points from two distinct clusters each. From the previous paragraph, given a non-center $s \in C_i$ for $i \neq x, y$, we know that p and s are centers for C_x and C_y. With an identical argument, given $t \in C_j$ for $j \neq x, y, i$, we can show that q and t are centers for C_x and C_y. It follows that S_x and S_y both contain points from at least two distinct clusters.

Now we finish the proof by showing that for each pair $u \in S_x$, $v \in S_y$, $\{c_\ell\}_{\ell=1}^k \cup \{u, v\}$ $(3,3)$-hits S. Given a non-center $s \in C_i$, without loss of generality $s \in S_x$, then there exists $j \neq i$ and $t \in C_j \cap S_x$. Then c_i, c_j, and u are $3r^*$ to s and c_i, c_x, and u are $3r^*$ to c_i. In the case where $i = x$, then c_i, c_j, and u are $3r^*$ to c_i. This concludes the proof. □

Now we move to the second half of the proof of Theorem 2.6.8. Recall that the proof from the previous section relied on a ranking argument, in which optimal clusters were mapped to their closest centers from the set C of $k+2$ points from the first half of the proof. This is the basis for the following fact.

Fact 2.6.12. *Given a k-center clustering instance (S, d) such that all optimal clusters have size $> 2\epsilon n$, and an α-perturbation d' of d, let $\mathcal{C}'$ denote the set of (α, ϵ)-PR clusters. For each $C_x \in \mathcal{C}'$, there exists a ranking $R_{x,d'}$ of S such that for any set of optimal centers C, the center for C_x is the highest-ranked point in $R_{x,d'}$.* [16]

Proof. Assume the lemma is false. Then there exists an (α, ϵ)-PR cluster C_i, two distinct points $u, v \in S$, and two sets of k centers C and C' both containing u and v, and both sets achieve the optimal score under an α-perturbation d', but u is the center for C_i in C while v is the center for C_i in C'. Then $\text{Vor}_C(u)$ is ϵ-close to C_i; similarly, $\text{Vor}_{C'}(v)$ is ϵ-close to C_i. This implies u is closer to all but ϵn points in C_i than v, and v is closer to all but ϵn points in C_i than u. Since $|C_i| > 2\epsilon n$, this causes a contradiction. □

We also define $R_{x,d',C} : C \to [n']$ as the ranking specific to C. Recall that our goal is to show a contradiction assuming two points from different PR clusters are close. From Lemma 2.6.6 and Lemma 2.6.11, we know there is a set of $k+2$ points, and any size k subset is optimal under a suitable perturbation. By Lemma 2.6.10, each size k subset must have a mapping from PR clusters to centers, and from Fact 2.6.12, these mappings are derived from a ranking of all possible center points by the PR clusters. In other words, each PR cluster C_x can rank all the points in S, so that for any set of optimal centers for an α-perturbation, the top-ranked center is the one whose

[16] Formally, for each $C_x \in \mathcal{C}'$, there exists a bijection $R_{x,d'} : S \to [n]$ such that for all sets of k centers C that achieve the optimal cost under d', then $c = \text{argmin}_{c' \in C} R_{x,d'}(c')$ if and only if $\text{Vor}_C(c)$ is ϵ-close to C_x.

cluster is ϵ-close to C_x. Now, using Fact 2.6.12, we can try to give a contradiction by showing that there is no set of rankings for the PR clusters that is consistent with all the optimal sets of centers guaranteed by Lemmas 2.6.6 and 2.6.11. The following lemma gives relationships among the possible rankings. These will be our main tools for contradicting PR and thus finishing the proof of Theorem 2.6.8.

Lemma 2.6.13. *Given a k-center clustering instance (S, d) such that all optimal clusters are size $> 2\epsilon n$, and given non-centers $p, q \in S$ such that $C = \{c_\ell\}_{\ell=1}^k \cup \{p, q\}$ $(3, 3)$-hits S, let the set $\mathcal{C}'$ denote the set of $(3, \epsilon)$-PR clusters. Define the 3-perturbation d' as in Lemma 2.6.6. The following are true.*

1. *Given $C_x \in \mathcal{C}'$ and C_i such that $i \neq x$, $R_{x,d'}(c_x) < R_{x,d'}(c_i)$.*
2. *There do not exist $s \in C$ and $C_x, C_y \in \mathcal{C}'$ such that $x \neq y$, and $R_{x,d',C}(s) + R_{y,d',C}(s) \leq 4$.*
3. *Given C_i and $C_x, C_y \in \mathcal{C}'$ such that $x \neq y \neq i$, if $R_{x,d',C}(c_i) \leq 3$, then $R_{y,d',C}(p) \geq 3$ and $R_{y,d',C}(q) \geq 3$.*

Proof. 1. By definition of the optimal clusters, for each $s \in C_x$, $d(c_x, s) < d(c_i, s)$, and therefore by construction, $d'(c_x, s) < d'(c_i, s)$. It follows that $R_{x,d'}(c_x) < R_{x,d'}(c_i)$.

2. Assume there exists $s \in C$ and $C_x, C_y \in \mathcal{C}'$ such that $R_{x,d',C}(s) + R_{y,d',C}(s) \leq 4$.

   Case 1: $R_{x,d',C}(s) = 1$ and $R_{y,d',C}(s) \leq 3$. Define u and v such that $R_{y,d',C}(u) = 1$ and $R_{y,d',C}(v) = 2$. (If u or v is equal to s, then redefine it to an arbitrary center in $C \setminus \{s, u, v\}$.) Consider the set of centers $C' = C \setminus \{u, v\}$ which is optimal under d' by Lemma 2.6.6. By Fact 2.6.12, s is the center for the majority of points in both C_x and C_y, causing a contradiction.

   Case 2: $R_{x,d',C}(s) = 2$ and $R_{y,d',C}(s) = 2$. Define u and v such that $R_{x,d',C}(u) = 1$ and $R_{y,d',C}(v) = 1$. (Again, if u or v is equal to s, then redefine it to an arbitrary center in $C \setminus \{s, u, v\}$.) Consider the set of centers $C' = C \setminus \{u, v\}$ which is optimal under d' by Lemma 2.6.6. However, by Fact 2.6.12, s is the center for the majority of points in both C_x and C_y, causing a contradiction.

3. Assume $R_{x,d',C}(c_i) \leq 3$.

   Case 1: $R_{x,d',C}(c_i) = 2$. Then by Lemma 2.6.13 part 1, $R_{x,d',C}(c_x) = 1$. Consider the set of centers $C' = C \setminus \{c_x, p\}$, which is optimal under d'. By Fact 2.6.12, $\text{Vor}_{C'}(c_i)$ must be ϵ-close to C_x. In particular, $\text{Vor}_{C'}(c_i)$ cannot contain more than ϵn points from C_i. But by definition, for all $j \neq i$ and $s \in C_i$, $d(c_i, s) < d(c_j, s)$. It follows that $\text{Vor}_{C'}(q)$ must contain all but ϵn points from C_i. Therefore, for all but ϵn points $s \in C_i$, for all j, $d'(q, s) < d'(c_j, s)$. If $R_{y,d',C}(q) \leq 2$, then C_y ranks c_y or p number one. Then for the set of centers $C' = C \setminus \{c_y, p\}$, $\text{Vor}_{C'}(q)$ contains more than ϵn points from C_y and C_i, contradicting the fact that C_y is $(3, \epsilon)$-PR. Therefore, $R_{y,d',C}(q) \geq 3$. The argument to show $R_{y,d',C}(p) \geq 3$ is symmetric.

   Case 2: $R_{x,d',C}(c_i) = 3$. If there exists $j \neq i, x$ such that $R_{x,d',C}(c_i) = 2$, then WLOG we are back in case 1. By Lemma 2.6.13 part 1, $R_{x,d',C}(c_x) \leq 2$. Then either p or q are

ranked top two, WLOG $R_{x,d',C}(p) \leq 2$. Consider the set $C' = C \setminus \{c_x, p\}$. Then as in the previous case, $\text{Vor}_{C'}(c_i)$ must be ϵ-close to C_x, implying for all but ϵn points $s \in C_i$, for all j, $d'(q,s) < d'(c_j,s)$. If $R_{y,d',C}(q) \leq 2$, again, C_y ranks c_y or p as number one. Let $C' = C \setminus \{c_y, p\}$, and then $\text{Vor}_{C'}(q)$ contains more than ϵn points from C_y and C_i, causing a contradiction. Furthermore, if $R_{y,d',C}(p) \leq 2$, then we arrive at a contradiction by Lemma 2.6.13 part 2.

□

We are almost ready to bring everything together to give a contradiction. Recall that Lemma 2.6.11 allows us to choose a pair (p,q) such that $\{c_\ell\}_{\ell=1}^k \cup \{p,q\}$ $(3,3)$-hits S. For an arbitrary choice of p and q, we may not end up with a contradiction. It turns out, we will need to make sure one of the points comes from a PR cluster, and is very high in the ranking list of its own cluster. This motivates the following fact, which is the final piece to the puzzle.

Fact 2.6.14. *Given a k-center clustering instance (S,d) such that all optimal clusters are size $> 2\epsilon n$, given an (α,ϵ)-PR cluster C_x, and given $i \neq x$, then there are fewer than ϵn points $s \in C_x$ such that $d(c_i,s) \leq \min(r^*, \alpha d(c_x,s))$.*

Proof. Assume the fact is false. Then let $B \subseteq C_x$ denote a set of size ϵn such that for all $s \in B$, $d(c_i,s) \leq \min(r^*, \alpha d(c_x,s))$. Construct the following perturbation d'. For all $s \in B$, set $d'(c_x,s) = \alpha d(c_x,s)$. For all other pairs s,t, set $d'(s,t) = d(s,t)$. This is clearly an α-perturbation by construction. Then the original set of optimal centers still achieves cost r^* under d' because for all $s \in B$, $d'(c_i,s) \leq r^*$. Clearly, the optimal cost under d' cannot be $< r^*$. It follows that the original set of optimal centers C is still optimal under d'. However, all points in B are no longer in $\text{Vor}_C(c_x)$ under d', contradicting the fact that C_x is (α,ϵ)-PR. □

Now we are ready to prove Theorem 2.6.8.

Proof of Theorem 2.6.8. Assume towards contradiction that there are two points at distance $\leq r^*$ from different $(3,\epsilon)$-PR clusters. Then by Lemma 2.6.11, there exists a partition S_1, S_2 of non-centers of S such that for all pairs $p \in S_1$, $q \in S_2$, $\{c_\ell\}_{\ell=1}^k \cup \{p,q\}$ $(3,3)$-hit S. Given three $(3,\epsilon)$-PR clusters C_x, C_y, and C_z, let c'_x, c'_y, and c'_z denote the optimal centers ranked highest by C_x, C_y, and C_z disregarding c_x, c_y, and c_z, respectively. Define $p = \text{argmin}_{s \in C_x} d(c_x,s)$, and WLOG let $p \in S_1$. Then pick an arbitrary point q from S_2, and define $C = \{c_\ell\}_{\ell=1}^k \cup \{p,q\}$. Define d' as in Lemma 2.6.6. We claim that $R_{x,d',C}(p) < R_{x,d',C}(c'_x)$: from Fact 2.6.14, there are fewer than ϵn points $s \in C_x$ such that $d(c'_x,s) \leq \min(r^*, 3d(c_x,s))$. Among each remaining point $s \in C_x$, we will show $d'(p,s) \leq d'(c'_x,s)$. Recall that $d(p,s) \leq d(p,c_x) + d(c_x,s) \leq 2r^*$, so $d'(p,s) = \min(3r^*, 3d(p,s))$. There are two cases to consider.

Case 1: $d(c'_x,s) > r^*$. Then by construction, $d'(c'_x,s) \geq 3r^*$, and so $d'(p,s) \leq d'(c'_x,s)$.

Case 2: $3d(c_x, s) < d(c'_x, s)$. Then

$$\begin{aligned} d'(p,s) &\leq 3d(p,s) \text{ by construction of } d' \\ &\leq 3(d(p,c_x) + d(c_x,s)) \text{ by triangle inequality} \\ &\leq 6d(c_x,s) \text{ by definition of } p \\ &\leq 2d(c'_x,s) \text{ by assumption} \\ &\leq \min(3r^*, 3d(c'_x,s)) \text{ by construction of } d' \\ &= d'(c'_x,s), \end{aligned}$$

and this proves our claim.

Because $R_{x,d',C}(p) < R_{x,d',C}(c'_x)$, it follows that either $R_{x,d',C}(p) \leq 2$ or $R_{x,d',C}(q) \leq 2$, since the top two can only be c_x, p, or q. The rest of the argument is broken up into cases.

Case 1: $R_{x,d',C}(c'_x) \leq 3$. From Lemma 2.6.13, then $R_{y,d',C}(p) \geq 3$ and $R_{y,d',C}(q) \geq 3$. It follows by process of elimination that $R_{y,d',C}(c_y) = 1$ and $R_{y,d',C}(c_{y'}) = 2$. Again by Lemma 2.6.13, $R_{x,d',C}(p) \geq 3$ and $R_{x,d',C}(q) \geq 3$, causing a contradiction.

Case 2: $R_{x,d',C}(c_{x'}) > 3$ and $R_{y,d',C}(c_{y'}) \leq 3$. Then $R_{x,d',C}(p) \leq 3$ and $R_{x,d',C}(q) \leq 3$. From Lemma 2.6.13, $R_{x,d',C}(p) \geq 3$ and $R_{x,d',C}(q) \geq 3$, therefore we have a contradiction. Note, the case where $R_{x,d',C}(c_{x'}) > 3$ and $R_{z,d',C}(c_{z'}) \leq 3$ is identical to this case.

Case 3: The final case is when $R_{x,d',C}(c_{x'}) > 3$, $R_{y,d',C}(c_{y'}) > 3$, and $R_{z,d',C}(c_{z'}) > 3$. So for each $i \in \{x, y, z\}$, the top three for C_i in C is a permutation of $\{c_i, p, q\}$. Then each $i \in \{x, y, z\}$ must rank p or q in the top two, so by the Pigeonhole Principle, either p or q is ranked top two by two different PR clusters, contradicting Lemma 2.6.13. This completes the proof. □

We note that Case 3 in Theorem 2.6.8 is the reason why we need to assume there are at least three $(3, \epsilon)$-PR clusters. If there are only two, C_x and C_y, it is possible that there exist $u \in C_x$, $v \in C_y$ such that $d(u, v) \leq r^*$. In this case, for p, q, d', and C as defined in the proof of Theorem 2.6.8, if C_x ranks c_x, p, q as its top three and C_y ranks c_y, q, p as its top three, then there is no contradiction.

2.6.3 Asymmetric k-center

Now we consider asymmetric k-center under $(3, \epsilon)$-PR. The asymmetric case is a more challenging setting, and our algorithm does not return the optimal solution, however, our algorithm outputs a clustering that is ϵ-close to the optimal solution.

Recall the definition of the symmetric set A from Section 2.3, $A = \{p \mid \forall q, d(q,p) \leq r^* \implies d(p,q) \leq r^*\}$, equivalently, the set of all CCV's. We might first ask whether A respects the structure of $\mathcal{OPT}$, as it did under 2-perturbation resilience. Namely, whether *Condition 1:* all optimal centers are in A, and *Condition 2:* $\arg\min_{q \in A} d(q,p) \in C_i \implies p \in C_i$ hold. In fact, we will show that neither conditions hold in the asymmetric case, but both conditions are only slightly violated.

Structure of optimal centers

First we give upper and lower bounds on the number of optimal centers in A, which will help us construct an algorithm for $(3, \epsilon)$-PR later on. We call a center c_i "bad" if it is not in the set A, i.e.,

$\exists q$ such that $d(q, c_i) \leq r^*$ but $d(c_i, q) > r^*$. First we give an example of a $(3, \epsilon)$-PR instance with at least one bad center, and then we show that all $(3, \epsilon)$-PR centers must have at most 6 bad centers.

Lemma 2.6.15. *For all $\alpha, n, k \geq 1$ such that $\frac{n}{k} \in \mathbb{N}$, there exists a clustering instance with one bad center satisfying $\left(\alpha, \frac{2}{n}\right)$-perturbation resilience.*

Proof. Given $\alpha, n, k \geq 1$, we construct a clustering instance such that all clusters are size $\frac{n}{k}$. Denote the clusters by $C_1, \ldots, C_k$ and the centers by $c_1, \ldots, c_k$. For each i, denote the non-centers in C_i by $p_{i,1}, \ldots, p_{i,L}$. Now we define the distances as follows. For convenience, set $L = \frac{n}{k} - 1$. For all $2 \leq i \leq k$ and $1 \leq j \leq L$, let $d(c_i, p_{i,j}) = 1$. For all $2 \leq i \leq k$, $1 \leq j, \ell \leq L$, let $d(p_{i,j}, p_{1,\ell}) = \frac{1}{\alpha}$ and $d(c_1, p_{1,\ell}) = \frac{1}{\alpha}$. Finally, let $d(p_{2,1}, c_1) = 1$. All other distances are the maximum allowed by the triangle inequality. In particular, the distance between two points p and q is set to infinity unless there exists a path from p to q with finite distance edges defined above. See Figure 2.8.

The optimal clusters and centers are $C_1, \ldots, C_k$ and $c_1, \ldots, c_k$, achieving a radius of 1, and c_1 is a bad center because $d(p_{2,1}, c_1) = 1$ but $d(c_1, p_{2,1}) = \infty$. It is left to show that this instance satisfies $(\alpha, \frac{2}{n})$-perturbation resilience. Given an arbitrary α-perturbation d', we must show that at most $\frac{2}{n} \cdot n = 2$ points switch clusters. By definition of an α-perturbation, for all p, q, we have $d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$ (recall that without loss of generality, an α-perturbation only inc Assume without loss of generality that for all p, q, $d(p, q) \leq d'(p, q)$ (i.e., d' only scales up the distances). The centers $c_2, \ldots, c_k$ ***must*** remain optimal centers under d', since for all $2 \leq i \leq k$, $d'(c_i, p_{i,1}) \leq \alpha$ and no other point $q \neq c_i, p_{i,1}$ satisfies $d(q, p_{i,1}) < \infty$. Now we must determine the final optimal center. Note that for all $2 \leq i, j \leq k$ and $1 \leq \ell, m \leq L$, we have

$$\begin{aligned} d'(p_{i,\ell}, p_{1,m}) &\leq \alpha d(p_{i,\ell}, p_{1,m}) \\ &< \alpha \cdot \frac{1}{\alpha} \\ &\leq d(c_j, p_{1,m}) \\ &\leq d'(c_j, p_{1,m}). \end{aligned}$$

Therefore, c_j cannot be a center for $p_{i,\ell}$, for all $2 \leq i, j \leq k$ and $1 \leq \ell \leq L$. Therefore, the final optimal center c under d' must be either c_1 or $p_{i,\ell}$ for $2 \leq i \leq k$ and $1 \leq \ell \leq L$. Furthermore, it follows that c's cluster at least contains $C_1 \setminus \{c_1\}$ and for each $2 \leq i \leq k$, c_i's cluster at least contains $C_i \setminus \{c\}$. Therefore, the optimal clustering under d' differs from $\mathcal{OPT}$ by at most two points. This concludes the proof. □

Now we show there are at most 6 bad centers for any asymmetric k-center instance satisfying $(3, \epsilon)$-PR.

Lemma 2.6.16. *Given a $(3, \epsilon)$-perturbation resilient asymmetric k-center instance such that all optimal clusters are size $> 2\epsilon n$, there are at most 6 bad centers, i.e., at most 6 centers c_i such that $\exists q$ with $d(q, c_i) \leq r^*$ and $d(c_i, q) > r^*$.*

Proof. Assume the lemma is false. By assumption, there exists a set B, $|B| \geq 7$, of centers c_i such that $\exists q$ with $d(q, c_i) \leq r^*$ and $d(c_i, q) > r^*$. The first step is use this set of bad centers to construct

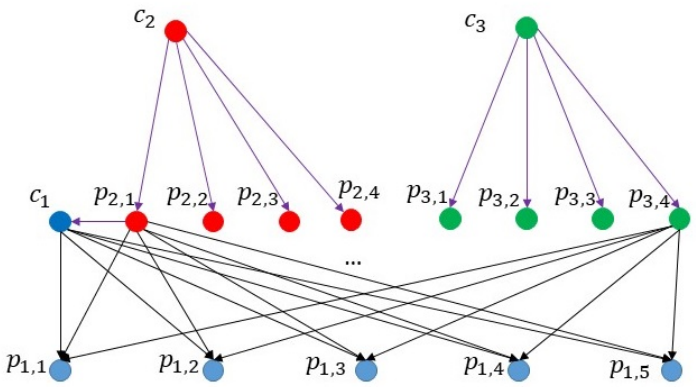


Figure 2.8: *A $(3,\epsilon)$-perturbation resilient asymmetric k-center instance with one bad center (c_y). The purple arrows are distance 1, and the black arrows are distance $\frac{1}{\alpha}$.*

a set C of $\leq k-3$ points which are $\leq 3r^*$ from every point in S. Once we find C, we will show how this set cannot exist under $(3,\epsilon)$-perturbation resilience, causing a contradiction.

Given a center $c_i \in B$, and q such that $d(q,c_i) \leq r^*$ and $d(c_i,q) > r^*$, note that $d(c_i,q) > r^*$ implies $q \notin C_i$. For each $c_i \in B$, define $a(i)$ as the center of q's cluster. Then $d(a(i),c_i) \leq d(a(i),q) + d(q,c_i) \leq 2r^*$ and so for all $p \in C_i$, we have $d(a(i),p) \leq d(a(i),c_i) + d(c_i,p) \leq 3r^*$. If for each $c_i \in B$, $a(i)$ is not in B, then we would be able to remove B from the set of optimal centers, and the remaining centers are still distance $3r^*$ from all points in S (finishing the first half of the proof). However, we need to consider the case where there exist centers c_i in B such that $a(i)$ is also in B. Our goal is to show there exists a subset $B' \subseteq B$ of size 3, such that for each $c_i \in B'$, $a(i) \notin B'$, therefore, the set of optimal centers without B' is still distance $3r^*$ from all points in S.

Construct a directed graph $G = (B,E)$ where $E = \{(c_i,c) \mid c = a(i)\}$. Then every point has out-degree ≤ 1. Finding B' corresponds to finding ≥ 3 points with no edges to one another, i.e., an independent set of G. Consider a connected component $G' = (V',E')$ of G. Since V' is connected, we have $|E'| \geq |V'| - 1$. Since every vertex has out-degree ≤ 1, $|E'| \leq |V'|$. Then we have two cases.

Case 1: $|E'| = |V'| - 1$. Then G' is a tree, and so there must exist an independent set of size $\left\lceil \frac{|V'|}{2} \right\rceil$.

Case 2: $|E'| = |V'|$. Then G' contains a cycle, and so there exists an independent set of size $\left\lfloor \frac{|V'|}{2} \right\rfloor$.

It follows that we can always find an independent set of size $\left\lfloor \frac{|V'|}{2} \right\rfloor$ for the entire graph G. For $|B| \geq 7$, there exists such a set B' of size ≥ 3. Then we have the property that $c_i \in B' \implies a(i) \notin B'$.

Now let $C = \{c_\ell\}_{\ell=1}^k \setminus B'$. By construction, B' is distance $\leq 3r^*$ to all points in S. Consider the following 3-perturbation d': increase all distances by a factor of 3, except $d(a(i),p)$, for i such that

$c_i \in B'$ and $p \in C_i$, which we increase to $\min(3r^*, 3d(a(i), p))$. Then by Lemma 2.2.8, the optimal radius is $3r^*$. Therefore, the set C achieves the optimal cost over d' even though $|C| \leq k-3$. Then we can pick any combination of 3 dummy centers, and they must all result in clusterings which are ϵ-close to $\mathcal{OPT}$. We will show this contradicts $(3, \epsilon)$-perturbation resilience.

We pick five arbitrary points $p_1, p_2, p_3, p_4, p_5 \in S \setminus C$, and define $C' = C \cup \{p_1, p_2, p_3, p_4, p_5\}$. From the above paragraph, each size 3 subset $P \subseteq \{p_1, p_2, p_3, p_4, p_5\}$ added to C will result in a set of optimal centers under d'. Then by Fact 2.6.2, each point in $C \cup P$ must be the center for the majority of points in exactly one cluster. To obtain a contradiction, we consider the ranking defined by Fact 2.6.7 of C' over d'.

We we start with a claim about the rankings: for each $c' \in C'$, for all pairs x, y such that $x \neq y$, if $\nexists c \in C$ such that $R_{x,d',C'}(c) < R_{x,d',C'}(c')$ or $R_{y,d',C'}(c) < R_{y,d',C'}(c')$, then $R_{x,d',C'}(c') + R_{y,d',C'}(c') \geq 5$. In words, there cannot be two clusters such that c' is ranked first among $C \cup \{c'\}$ and top two (or first and third) among C' for both clusters. Assume this is false. Then there exist $x \neq y$ such that $R_{x,d',C'}(c') + R_{y,d',C'}(c') \leq 4$, so there are at most two total points ranked above c' in $R_{x,d',C'}$ and $R_{y,d',C'}$, and these points must be from the set $\{p_1, p_2, p_3, p_4, p_5\}$. Without loss of generality, denote these points by p and p' (if there are one or zero points ranked above c', let one or both of p and p' be arbitrary). Then consider the set of centers $C' \setminus \{p, p'\}$ which is size k and must be optimal under d' as described earlier. However, the partitioning is not ϵ-close to $\mathcal{OPT}$, since c' is the best center (ranked 1) for both C_x and C_y. This completes the proof of the claim.

Now consider the set $D = \{c_i \in C \mid \exists x \text{ s.t. } R_{x,d',C'}(c_i) = 1\}$, i.e., the set of points in C which are ranked 1 for some cluster. Denote $m = (k-3) - |D|$, which is the number of points in C which are *not* ranked 1 for any cluster. By the claim and since $|C| = k-3$, there are exactly $m+3$ clusters whose top-ranked point is not in C. Given one such cluster C_x, again by the claim, the top two ranked points must not be from the set D. Therefore, there are $2(m+3)$ slots that must be filled by $m+5$ points, so (for all $m \geq 0$) by the Pigeonhole Principle, there must exist a point $p \in C'$ ranked in the top two by two different clusters. This directly contradicts the claim, so we have a contradiction which completes the proof. □

Algorithm under $(3, \epsilon)$-PR

From the previous lemma, we know that at most a constant number of centers are bad. Essentially, our algorithm runs a symmetric 2-approximation algorithm on A, for all $k-6 \leq k' \leq k$, to find a 2-approximation for the clusters in A. For instance, iteratively pick an unmarked point, and mark all points distance $2r^*$ away from it [Hochbaum and Shmoys, 1985]. Then we use brute force to find the remaining 6 centers, which will give us a 3-approximation for the entire point set. Under $(3, \epsilon)$-perturbation resilience, this 3-approximation must be ϵ-close to $\mathcal{OPT}$. We are not able to output $\mathcal{OPT}$ exactly, since Condition 2 may not be satisfied for up to ϵn points.

Theorem 2.6.17. *Algorithm 5 runs in polynomial time and outputs a clustering that is ϵ-close to $\mathcal{OPT}$, for $(3, \epsilon)$-perturbation resilient asymmetric k-center instances s.t. all optimal clusters are size $> 2\epsilon n$.*

Proof. We define three types of clusters. A cluster C_i is green if $c_i \in A$, it is yellow if $c_i \notin A$ but $C_i \cap A \neq \emptyset$, and it is red if $C_i \cap A = \emptyset$. Denote the number of yellow clusters by y, and the number of red clusters by x. From Lemma 2.6.16, we know that $x + y \leq 6$. The symmetric

Algorithm 5 $(3, \epsilon)$-PERTURBATION RESILIENT ASYMMETRIC k-CENTER

Input: Asymmetric k-center instance (S, d), r^* (or try all possible candidates).

- Build set $A = \{p \mid \forall q, d(q, p) \leq r^* \implies d(p, q) \leq r^*\}$.
- Create the threshold graph $G = (A, E)$ where $E = \{(u, v) \mid d(u, v) \leq r^*\}$. Define a new symmetric k-center instance (S, A, d') where $d'(u, v) = dist_G(u, v)$.
- For all $k - 6 \leq k' \leq k$, run a symmetric k-center 2-approximation algorithm on (S, A, d'). Break if the output is a set of centers C achieving cost $\leq 2r^*$.
- For all $C' \subseteq C$ of size $k - 6$ and $S' \subseteq S$ of size 6, return if cost$(C' \cup S') \leq 3r^*$.

Output: Voronoi tiling $G_1, \ldots, G_k$ using $C' \cup S'$ as the centers.

k-center instance (S, A, d') constructed in step 2 of the algorithm is a subset of an instance with $k - x$ optimal clusters of cost r^*, so the $(k - x)$-center cost of (S, A, d') is at most r^*. Therefore, step 3 will return a set of centers achieving cost $\leq 2r^*$ for some $k' \leq k - x$. By definition of green clusters, we know that $k - x - y$ clusters have their optimal center in A. For each green cluster C_i, let $c(i) \in C$ denote the center which is distance $\leq 2r^*$ to c_i (if there is more than one point in C, denote $c(i)$ by one of them arbitrarily). Let $C' = \{c(i) \mid C_i \text{ is green}\}$, and $|C'| \leq k - x - y$. Then the set $C' \cup \{c_x \mid x \text{ is not green}\}$ is cost $\leq 3r^*$, and the algorithm is guaranteed to encounter this set in the final step.

Finally, we explain why $C' \cup B$ must be ϵ-close to $\mathcal{OPT}$. Create a 3-perturbation in which we increase all distances by 3, except for the distances from $C' \cup B$ to all points in their Voronoi tile, which we increase up to $3r^*$. Then, the optimal score is $3r^*$ by Lemma 2.4.1, and $C' \cup B$ achieves this score. Therefore, by $(3, \epsilon)$-perturbation resilience, the Voronoi tiling of $C' \cup B$ must be ϵ-close to $\mathcal{OPT}$. This completes the proof. □

2.6.4 APX-Hardness under perturbation resilience

Now we show hardness of approximation even when it is guaranteed the clustering satisfies (α, ϵ) -approximation stability for $\alpha \geq 1$ and $\epsilon > 0$. The hardness is based on a reduction from the general clustering instances, so the APX-hardness constants match the non-stable APX-hardness results. This shows the condition on the cluster sizes in Theorem 2.6.1 is tight. [17]

Theorem 2.6.18. *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 3.94, even when it is guaranteed the instance satisfies (α, ϵ)-approximation stability.*

Proof. Given $\alpha \geq 1$, $\epsilon > 0$, assume there exists a β-approximation algorithm $\mathcal{A}$ for k-median under (α, ϵ)-approximation stability. We will show a reduction to k-median without approximation stability. Given a k-median clustering instance (S, d) of size n, we will create a new instance (S', d') for $k' = k + n/\epsilon$ with size $n' = n/\epsilon$ as follows. First, set $S' = S$ and $d' = d$, and then add n/ϵ new points to S', such that their distance to every other point is $2\alpha n \max_{u,v \in S} d(u, v)$. Let

[17] In fact, this hardness holds even under the strictly stronger notion of *approximation stability* [Balcan et al., 2013a], therefore, it generalizes a hardness result from [Balcan et al., 2013a].

$\mathcal{OPT}$ denote the optimal solution of (S, d). Then the optimal solution to (S', d') is to use $\mathcal{OPT}$ for the vertices in S, and make each of the n/ϵ added points a center. Note that the cost of $\mathcal{OPT}$ and the optimal clustering for (S', d') are identical, since the added points are distance 0 to their center. Given a clustering $\mathcal{C}$ on (S, d), let $\mathcal{C}'$ denote the clustering of (S', d') that clusters S as in $\mathcal{C}$, and then adds n/ϵ extra centers on each of the added points. Then the cost of $\mathcal{C}$ and $\mathcal{C}'$ are the same, so it follows that $\mathcal{C}$ is a β-approximation to (S, d) if and only if $\mathcal{C}'$ is a β-approximation to (S', d'). Next, we claim that (S', d') satisfies (α, ϵ)-approximation stability. Given a clustering $\mathcal{C}'$ which is an α-approximation to (S', d'), then there must be a center located at all n/ϵ of the added points, otherwise the cost of $\mathcal{C}'$ would be $> \alpha\mathcal{OPT}$. Therefore, $\mathcal{C}'$ agrees with the optimal solution on all points except for S, therefore, $\mathcal{C}'$ must be ϵ-close to the optimal solution. Now that we have established a reduction, the theorem follows from hardness of 1.73-approximation for k-median [Jain et al., 2002]. The proofs for k-center and k-means are identical, using hardness from [Gonzalez, 1985] and [Lee et al., 2017], respectively. □

Chapter 3

Data-Driven Clustering

3.1 Introduction

In this chapter, we take a different approach to beyond worst-case analysis. Clustering arises 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 [2016] 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 [Rice, 1976] and has led to breakthroughs in diverse fields including combinatorial auctions [Leyton-Brown et al., 2009], scientific computing [Demmel et al., 2005], vehicle routing [Caseau et al., 1999], and SAT [Xu et al., 2008].

The most popular method in practice for clustering is local search, where we start with k centers and iteratively make incremental improvements until a local optimum is reached. For example, Lloyd's method (sometimes called k-means) [Lloyd, 1982] and k-medoids [Friedman et al., 2001, Cohen et al., 2016] are two popular local search algorithms. Another widely used paradigm for clustering algorithms is the agglomerative algorithm paradigm, which includes common linkage-

based methods such as single-, average-, and complete-linkage.

In the algorithm configuration framework, we study both types of algorithmic families for clustering: agglomerative clustering algorithms followed by a dynamic programming step, and Lloyd's local search algorithms with an initialization step.

There are multiple decisions an algorithm designer must make when using a Lloyd's local search algorithm or an agglomerative clustering algorithm. In local search, the algorithm designer must decide how to seed local search, e.g., how the algorithm chooses the k initial centers. There is a large body of work on seeding algorithms, since the initial choice of centers can have a large effect on the quality of the outputted clustering [Higgs et al., 1997, Pena et al., 1999, Arai and Barakbah, 2007]. The best seeding method often depends on the specific application at hand. For example, a "typical problem instance" in one setting may have significantly different properties from that in another, causing some seeding methods to perform better than others. Second, the algorithm designer must decide on an objective function for the local search phase (resp. dynamic programming phase) which could be k-means, k-median, etc. For some applications, there is an obvious choice. For instance, if the application is Wi-Fi hotspot location, then the explicit goal is to minimize the k-center objective function. For many other applications such as clustering communities in a social network, the goal is to find clusters which are close to an unknown target clustering, and we may use an objective function for local search in the hopes that approximately minimizing the chosen objective will produce clusterings which are close to matching the target clustering (in terms of the number of misclassified points). As before, the best objective function for local search may depend on the specific application.

In this chapter, we show positive theoretical results for learning the best clustering procedures over a large family of algorithms. We take a transfer learning approach where we assume there is an unknown distribution over problem instances corresponding to our application, and the goal is to use experience from the early instances to perform well on the later instances. For example, if our application is clustering facilities in a city, we would look at a sample of cities with existing optimally-placed facilities, and use this information to find the empirically best algorithm from an infinite family, and we use this algorithm to cluster facilities in new cities. This is similar in spirit to the *learning to learn* setting of Baxter [1997], where the goal is to learn the best concept class from a family of concept classes, for an unknown distribution.

Problem description. In the algorithm configuration framework, 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 $\mathcal{D}$. We derive our guarantees by analyzing the pseudo-dimension of the algorithm classes we study [Pollard, 1984, 1990, Anthony and Bartlett, 2009]). We then use the structure of the problem to provide efficient algorithms for most of the classes we study.

3.1.1 Results and techniques

Clustering by agglomerative algorithms with dynamic programming. 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.

For the linkage-based procedure, we consider several parameterized agglomerative procedures which induce a spectrum of algorithms interpolating between the popular single-, average-, and complete-linkage procedures, which are prevalent in practice [Awasthi et al., 2014, Saeed et al., 2003, White et al., 2010] and known to perform nearly optimally in many settings [Awasthi et al., 2012, Balcan et al., 2016, Balcan and Liang, 2016, Grosswendt and Roeglin, 2015]. For the dynamic programming step, we study an infinite class of objectives which include the standard k-means, k-median, and k-center objectives, common in applications such as information retrieval [Can, 1993, Charikar et al., 1997]. We show how to learn the best agglomerative algorithm and pruning objective function pair, thus extending our work to multiparameter algorithms. We provide tight pseudo-dimension bounds, ranging from $\Theta(\log n)$ for simpler algorithm classes to $\Theta(n)$ for more complex algorithm classes, so our learning algorithms are sample efficient.

(α, β)-Lloyds++ We define an infinite family of algorithms generalizing Lloyd's method, with two parameters α and β. Our algorithms have two phases, a seeding phase to find k initial centers (parameterized by α), and a local search phase which uses Lloyd's method to converge to a local optimum (parameterized by β). In the seeding phase, each point v is sampled with probability proportional to $d_{\min}(v, C)^\alpha$, where C is the set of centers chosen so far and $d_{\min}(v, C) = \min_{c \in C} d(v, c)$. Then Lloyd's method is used to converge to a local minima for the ℓ_β objective. By ranging $\alpha \in [0, \infty) \cup \{\infty\}$ and $\beta \in [1, \infty) \cup \{\infty\}$, we define our infinite family of algorithms which we call (α, β)-Lloyds++. Setting $\alpha = \beta = 2$ corresponds to the k-means++ algorithm [Arthur and Vassilvitskii, 2007]. The seeding phase is a spectrum between random seeding ($\alpha = 0$), and farthest-first traversal [Gonzalez, 1985, Dasgupta and Long, 2005] ($\alpha = \infty$), and the Lloyd's step is able to optimize over common objectives including k-median ($\beta = 1$), k-means ($\beta = 2$), and k-center ($\beta = \infty$). We design efficient learning algorithms which receive samples from an application-specific distribution over clustering instances and learn a near-optimal clustering algorithm from our family.

In Section 3.4, we prove that $O\left(\frac{1}{\epsilon^2} \min(T, k) \log n\right)$ samples are sufficient to guarantee the empirically optimal parameters $(\hat{\alpha}, \hat{\beta})$ have expected cost at most ϵ higher than the optimal parameters (α^*, β^*) over the distribution, with high probability over the random sample, where n is the size of the clustering instances and T is the maximum number of Lloyd's iterations. The key challenge is that for any clustering instance, the cost of the outputted clustering is not even a continuous function in α or β since a slight tweak in the parameters may lead to a completely different run of the algorithm. We overcome this obstacle by showing a strong bound on the expected number of discontinuities of the cost function, which requires a delicate reasoning about the structure of the "decision points" in the execution of the algorithm; in other words, for a given clustering instance, we must reason about the total number of outcomes the algorithm can produce over the full range

of parameters. This allows us to use Rademacher complexity, a distribution-specific technique for achieving uniform convergence.

Next, we complement our sample complexity result with a computational efficiency result. Specifically, we give a novel meta-algorithm which efficiently finds a near-optimal value $\hat{\alpha}$ with high probability. The high-level idea of our algorithm is to run depth-first-search over the "execution tree" of the algorithm, where a node in the tree represents a state of the algorithm, and edges represent a decision point. A key step in our meta-algorithm is to iteratively solve for the decision points of the algorithm, which itself is nontrivial since the equations governing the decision points do not have a closed-form solution. We show the equations have a certain structure which allows us to binary search through the range of parameters to find the decision points. Our algorithm has been shown to perform favorably on a number of different real-world and synthetic datasets including MNIST, Cifar10, CNAE-9, and mixtures of Gaussians [Balcan et al., 2018].

Key challenges. One of the key challenges in analyzing the pseudo-dimension of the algorithm classes we study 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 of a clustering algorithm 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 algorithm analyses require 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 analyses allows us to develop many computationally efficient meta-algorithms for algorithm configuration due to the related concept of *shattering*. A constrained pseudo-dimension of $O(\log n)$ often implies a small search space of $2^{O(\log n)} = O(n)$ in which the meta-algorithm will uncover a nearly optimal configuration.

We develop techniques for analyzing randomized algorithms, whereas the algorithms analyzed in the previous work were deterministic. We also provide the first pseudo-dimension lower bounds in this line of work, which require an involved analysis of each algorithm family's performance on carefully constructed instances. Our lower bounds are somewhat counterintuitive, since for several of the classes we study, they are of the order $\Omega(\log n)$, even if the corresponding classes of algorithms are defined by a single real-valued parameter.

3.1.2 Related work

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 [Awasthi et al., 2014], software clustering [Saeed et al., 2003], and bioinformatic datasets [White et al., 2010]. 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 [Awasthi et al., 2012], Balcan et al. show that single-linkage returns a near-optimal clustering for k-center under 2-perturbation resilience or $(3, \epsilon)$-perturbation resilience [Balcan et al., 2016], 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 [Balcan and Liang, 2016]. 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.

Lloyd's algorithm and farthest-first traversal The iterative local search method for clustering, known as Lloyd's algorithm or sometimes called k-means, is one of the most popular algorithms for k-means clustering [Lloyd, 1982], and improvements are still being found [Max, 1960, MacQueen et al., 1967, Dempster et al., 1977, Pelleg and Moore, 1999, Kanungo et al., 2002, Kaufman and Rousseeuw, 2009]. The worst-case runtime of Lloyd's method is exponential [Arthur and Vassilvitskii, 2006] even in $\mathbb{R}^2$ [Vattani, 2011], however, it converges very quickly in practice [Har-Peled and Sadri, 2005], and the smoothed complexity is polynomial Arthur et al. [2011]. Many different initialization approaches have been proposed [Higgs et al., 1997, Pena et al., 1999, Arai and Barakbah, 2007]. When using d^2-sampling to find the initial k centers, the algorithm is known as k-means++, and the approximation guarantee is provably $O(\log k)$ [Arthur and Vassilvitskii, 2007]. If the data satisfies a natural stability condition, k-means++ returns a near-optimal clustering [Ostrovsky et al., 2012].

The farthest-first traversal algorithm is an iterative method to find k centers, and it was shown to give a 2-approximation algorithm for k-center [Gonzalez, 1985], and an 8-approximation for hierarchical k-center [Dasgupta and Long, 2005].

Learning to Learn Application-specific algorithm configuration has been used in the artifical intelligence community for decades [Rice, 1976]. For example, Leyton-Brown et al. used algorithm configuration in designing combinatorial auctions [Leyton-Brown et al., 2009], Demmel et al. used this technique for scientific computing [Demmel et al., 2005], Caseau et al. studied the problem for vehicle routing [Caseau et al., 1999], and Xu et al. used this model to design a portfolio-based algorithm for SAT [Xu et al., 2008]. Gupta and Roughgarden were the first to theoretically introduce algorithm configuration as a formal learning framework [Gupta and Roughgarden, 2016]. 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.

There are several related models for learning the best representation and transfer learning for clustering. Ashtiani and Ben-David show how to use a small labeled fraction of the dataset to learn a metric embedding of the entire dataset such that k-means performs well [Ashtiani and Ben-David, 2015]. Baxter studied the task of choosing the right concept class for a distribution over an

environment of supervised learning tasks [Baxter, 1997]. Specifically, the goal is to learn the best hypothesis space from a family of hypothesis spaces. Our problem is similar in spirit to this model, where we study environments of unsupervised learning tasks instead of supervised learning tasks.

There are a few models for the question of finding the best clustering algorithm to use on a single instance, given a small amount of expert advice. Ackerman et al. (building off of the famous clustering impossibility result of Kleinberg [2003]) study the problem of taxonomizing clusering algorithmic paradigms, by using a list of abstract properties of clustering functions [Ackerman et al., 2010]. In their work, the goal is for a user to choose a clustering algorithm based on the specific properties which are important for her application.

Another related area is the problem of unsupervised domain adaption. In this problem, the machine learning algorithm has access to a labeled training dataset, and an unlabeled target dataset over a different distribution. The goal is to find an accurate classifier over the target dataset, while only training on the training distribution [Sener et al., 2016, Ganin and Lempitsky, 2015, Tzeng et al., 2014].

There has been more research on related questions for transfer learning on unlabeled data and unsupervised tasks. Raina et al. study transfer learning using unlabeled data, to a supervised learning task [Raina et al., 2007]. Jiang and Chung, and Yang et al. study transfer learning for clustering, in which a clustering algorithm has access to unlabeled data, and uses it to better cluster a related problem instance [Yang et al., 2009, Jiang and Chung, 2012]. This setting is a bit different from ours, since we assume we have access to the target clustering from each training instance, but we tackle the harder question of finding the best clustering objective.

3.2 Preliminaries

Problem description. Recall the model from the previous section, which we state again for convenience. 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 (Gupta and Roughgarden [2016])**.** *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 func-

tion $\texttt{cost}(h, x)$ 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 $S = \{x_1, x_2, \ldots x_m\}$ is shattered by the function class $\mathcal{H}$, if there exist real-valued witnesses $r_1, \ldots, r_m$ 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 $\texttt{cost}\,(h_T, x_i) \leq r_i$ 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 $\mathcal{H}$.

Next, we provide the definition of pseudo-dimension in the context of algorithm classes. 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 $\mathcal{H}_{\mathcal{A}} = \{\texttt{cost}(h, \cdot) : \mathcal{X} \to [0, H] \mid h \in \mathcal{A}\}$. Recall that a finite subset of problem instances $S = \{x_1, x_2, \ldots x_m\}$ is shattered by the function class $\mathcal{H}$, if there exist real-valued witnesses $r_1, \ldots, r_m$ 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 $\texttt{cost}\,(h_T, x_i) \leq r_i$ 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 [Dudley, 1967]: for every distribution $\mathcal{D}$ over $\mathcal{X}$, every $\epsilon > 0$, and every $\delta \in (0, 1]$,

$$m \geq c \left(\frac{H}{\epsilon}\right)^2 \left(Pdim(\mathcal{H}) + \log \frac{1}{\delta}\right)$$

for a suitable constant c (independent of all other parameters), then with probability at least $1 - \delta$ over m samples $x_1, \ldots, x_m \sim \mathcal{D}$,

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

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.

3.3 Agglomerative algorithms with dynamic programming

We begin with an overview of agglomerative algorithms with dynamic programming, which include many widely-studied clustering algorithms, and then we define several parameterized classes of such algorithms. As in the previous section, we prove it is possible to learn the optimal algorithm from a fixed class for a specific application, and for many of the classes we analyze, this procedure is computationally efficient and sample efficient. We focus on agglomerative algorithms with dynamic programming for *clustering* problems. A clustering instance $\mathcal{V} = (V, d)$ consists of a set V of n points and a distance metric $d : V \times V \to \mathbb{R}_{\geq 0}$ 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. 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 $\mathbf{c} = \{c_1, \ldots, c_k\} \subseteq V$ which we call centers, as well as a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ 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]

Next, we define agglomerative clustering algorithms with dynamic programming, which are prevalent in practice [Awasthi et al., 2014, Saeed et al., 2003, White et al., 2010] and enjoy strong theoretical guarantees in a variety of settings [Awasthi et al., 2012, Balcan et al., 2016, Balcan and Liang, 2016, Grosswendt and Roeglin, 2015]. 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. Algorithm 6 details how the merge function and pruning function work together to form an agglomerative clustering algorithm with dynamic programming. In the dynamic programming step, to find the 1-pruning of any node T, we only need to find the best center $c \in T$. When $k' > 1$, we recursively find the best k'-pruning of T by considering different combinations of the best i'-pruning of the left child T_L and the best $(k' - i')$-pruning of the right child T_R for $i' \in \{1, \ldots, k-1\}$ and choosing the best combination.

Pictorially, Figure 3.1 depicts an array of available choices when designing an agglomerative clustering algorithm with dynamic programming. Each path in the chart corresponds to an alternative choice of a merging function ξ and pruning function Ψ. The algorithm designer's goal is to determine the path that is optimal for her specific application domain.

Our results hold even when there is a fixed preprocessing step that precedes the agglomerative merge step (as long as it is independent of ξ and Ψ), therefore our analysis carries over to algorithms such as in [Balcan and Liang, 2016].

[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, Gupta and Tangwongsan [2008] provide an $O(p)$ approximation algorithm when $p < \log n$ and Bateni et al. [2014a] study distributed clustering algorithms.

Algorithm 6 Agglomerative algorithm with dynamic programming

Input: Clustering instance $\mathcal{V} = (V, d)$, merge function ξ, pruning function Ψ.

1: **Agglomerative merge step to build a cluster tree $\mathcal{T}$ according to ξ:**

- Start with n singleton sets $\{v\}$ for each $v \in V$.
- Iteratively merge the two sets A and B which minimize $\xi(A, B)$ until a single set remains.
- Let $\mathcal{T}$ denote the cluster tree corresponding to the sequence of merges.

2: **Dynamic programming to find the k-pruning of $\mathcal{T}$ minimizing Ψ :**

- For each node T, find the best k'-pruning of the subtree rooted at T in $\mathcal{T}$, denoted by $(\mathcal{C}_{T,k'}, \mathbf{c}_{T,k'})$ according to following dynamic programming recursion:

$$\Psi\left(\mathcal{C}_{T,k'}, \mathbf{c}_{T,k'}\right) = \begin{cases} \min_{c \in T} \Psi\left(\{T\}, c\right) & \text{if } k' = 1, \\ \min_{i' \in [k'-1]} \Psi\left(\mathcal{C}_{T_L,i'} \cup \mathcal{C}_{T_R,k'-i'}, \mathbf{c}_{T_L,i'} \cup \mathbf{c}_{T_R,k'-i'}\right) & \text{otherwise.} \end{cases}$$

where T_L and T_R denote the left and right children of T, respectively.

Output: The best k-pruning of the root node T_{root} of $\mathcal{T}$.

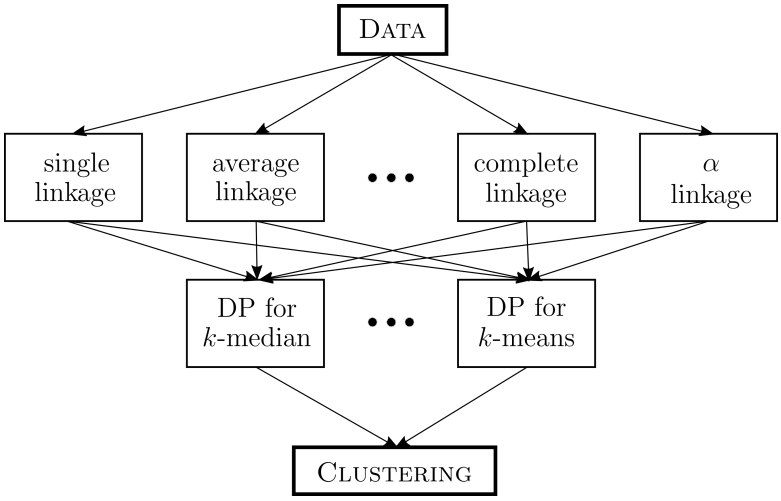


Figure 3.1: A schematic for a class of agglomerative clustering algorithms with dynamic programming.

3.3.1 Definition of algorithm classes

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 $\mathcal{A}_2$, depends on all pairwise distances between A and B. 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 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.

Recall that for a given class of merge functions and a `cost` function (a generic clustering objective `clus`), our goal is to learn a near-optimal value of α in expectation over an unknown distribution of clustering instances. One might wonder if there is some α that is optimal across all instances, which would preclude the need for a learning algorithm. In Theorem 3.3.1, we prove that this is not the case; for each $p \in [1, \infty) \cup \{\infty\}$ and $i \in \{1,2,3\}$, given any α, there exists a distribution over clustering instances for which $\mathcal{A}_i(\alpha)$ is the best algorithm in $\mathcal{A}_i$ with respect to $\Phi^{(p)}$. Crucially, this means that even if the algorithm designer sets p to be 1, 2, or ∞ as is typical in practice, the optimal choice of the tunable parameter α could be any real value. The optimal value of α depends on the underlying, unknown distribution, and must be learned, no matter the value of p.

To formally describe this result, we set up new notation. Let $\mathbb{V}$ denote the set of all clustering instances over at most n points. With a slight abuse of notation, we will use $\texttt{clus}_{(\mathcal{A}_i(\alpha),\Psi)}(\mathcal{V})$ 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 $\mathbb{E}_{\mathcal{V}\sim\mathcal{D}}\left[\texttt{clus}^{(p)}_{\mathcal{A}_i(\alpha)}(\mathcal{V})\right] < \mathbb{E}_{\mathcal{V}\sim\mathcal{D}}\left[\texttt{clus}^{(p)}_{\mathcal{A}_i(\alpha')}(\mathcal{V})\right]$ for all permissible values of $\alpha' \neq \alpha$ for $\mathcal{A}_i$.*

Proof. We give a general proof for all three classes $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. We will point out a few places in the proof where the details for $b = 1, 2, 3$ are different, but the general structure of the argument

is the same. For each value of b, we construct a single clustering instance $\mathcal{V} = (V, d)$ that has the desired property; the distribution $\mathcal{D}$ is merely the single clustering instance with probability 1.

Consider some permissible value of α, denoted α^*. Set $k = 4$ and $n = 210$. The clustering instance consists of two well-separated 'gadgets' of two clusters each. The class $\mathcal{A}_b$ results in different 2-clusterings of the first gadget depending on whether $\alpha \leq \alpha^*$ or not. Similarly, $\mathcal{A}_b$ results in different 2-clusterings of the second gadget depending on whether $\alpha \geq \alpha^*$ or not. By ensuring that for the first gadget $\alpha \leq \alpha^*$ results in the lowest cost 2-clustering, and for the second gadget $\alpha \geq \alpha^*$ results in the lowest cost 2-clustering, we ensure that $\alpha = \alpha^*$ is the optimal parameter overall.

The first gadget is as follows. We define five points a_1, b_1, c_1, x_1 and y_1. For the sake of convenience, we will group the remaining points into four sets A_1, B_1, X_1, and Y_1 each containing 25 points. We set the distances as follows: $d(a_1, b_1) = d(x_1, y_1) = 1$, $d(a_1, c_1) = 1.1$, and $d(b_1, c_1) = 1.2$. For $a \in A_1 \cup B_1$, $d(c_1, a) = 1.51$ and $d(a_1, a) = d(b_1, a) = 1.6$. For $x \in X_1 \cup Y_1$, $d(x_1, x) = d(y_1, x) = 1.6$. For $a \in A_1$, $b \in B_1$, $x \in X_1$, and $y \in Y_1$, $d(a, b) = d(x, y) = 1.6$. We also define special points $x_1^* \in X_1$ and $y_1^* \in Y_1$, which have the same distances as the rest of the points in X_1 and Y_1 respectively, except that $d(x_1, x_1^*) = 1.51$ and $d(y_1, y_1^*) = 1.51$. If two points p and q belong to the same set (A_1, B_1, X_1, or Y_1), then $d(p, q) = 1.5$.

The distances $d(x_1, c_1)$ and $d(y_1, c_1)$ are defined in terms of b and α^*, but they will always be between 1.1 and 1.2. For $b = 1$, we set $d(x_1, c_1) = d(y_1, c_1) = 1.2 - .1 \cdot \alpha^*$. For $b = 2$ and $b = 3$, $d(x_1, c_1) = d(y_1, c_1) = ((1.1^{\alpha^*} + 1.2^{\alpha^*})/2)^{\frac{1}{\alpha^*}}$.

So far, all of the distances we have defined are in $[1, 2]$, therefore they trivially satisfy the triangle inequality. We set all of the rest of the distances to be the maximum distances allowed under the triangle inequality. Therefore, the triangle inequality holds over the entire metric.

Now, let us analyze the merges caused by $\mathcal{A}_b(\alpha)$ for various values of α. Regardless of the values of α and b, since the distances between the first five points are the smallest, merges will occur over these initially. In particular, regardless of α and b, a_1 is merged with b_1, and x_1 with y_1. Next, by a simple calculation, if $\alpha \leq \alpha^*$, then c_1 merges with $a_1 \cup b_1$. If $\alpha > \alpha^*$, then c_1 merges with $x_1 \cup y_1$. Denote the set containing a_1 and b_1 by A_1', and denote the set containing x_1 and y_1 by X_1' (one of these sets will also contain c_1). Between A_1' and X_1', the minimum distance is $\geq 1.1 + 1.1 \geq 2.2$. All other subsequent merges (except for the very last merge) will involve all distances smaller than 2.2, so we never need to consider A_1' merging to X_1'.

The next smallest distances are all 1.5, so all points in A_1 will merge together, and similarly for B_1, X_1, and Y_1. At this point, the algorithm has created six sets: A_1', X_1', A_1, B_1, X_1, and Y_1. We claim that if $\alpha \leq \alpha^*$, A_1' will merge to A_1 and B_1, and X_1' will merge to X_1 and Y_1. This is because the maximum distance between sets in each of these merges is 1.6, whereas the minimum distance between $\{A_1', A_1, B_1\}$ and $\{X_1', X_1, Y_1\}$ is ≥ 2.2. Therefore, for all three values of b, the claim holds true.

Next we claim that the 2-clustering cost of gadget 1 will be lowest for clusters $A_1' \cup A_1 \cup B_1\}$ and $X_1' \cup X_1 \cup Y_1$ and when $c_1 \in A_1'$, i.e., when $\alpha \leq \alpha^*$. Clearly, since the distances within $A_1' \cup A_1 \cup B_1$ and $X_1' \cup X_1 \cup Y_1$ are much less than the distances across these sets, the best 2-clustering is $A_1' \cup A_1 \cup B_1$ and $X_1' \cup X_1 \cup Y_1$ (with all points at distance ≤ 1.6 to their center). We proved this will be a pruning of the tree when $\alpha \leq \alpha^*$. Therefore, we must argue the cost

of this 2-clustering is lowest when $c_1 \in A_1'$. The idea is that c_1 can act as a very good center for $A_1' \cup A_1 \cup B_1$. But if $c_1 \in X_1'$, then the best center for $A_1' \cup A_1 \cup B_1$ will be an arbitrary point in $A_1 \cup B_1$. The cost in the first case is $1.51^p \cdot 50 + 1.1^p + 1.2^p$. The cost in the second case is $1.5^p \cdot 24 + 1.6^p \cdot 27$.

For $X_1' \cup X_1 \cup Y_1$, the center does not change depending on α (x_1^* and y_1^* tie for the best center), so the only difference in the cost is whether or not to include c_1. If $\alpha \leq \alpha^*$, then the cost is $1.5^p \cdot 24 + 1.51^p + 1.6^p \cdot 26$, otherwise the cost is $1.5^p \cdot 24 + 1.51^p + 1.6^p \cdot 26 + (1.6 + 1.2 - 0.1\alpha^*)^p$.

Putting it all together, if $\alpha \leq \alpha^*$, the cost is $1.51^p \cdot 50 + 1.1^p + 1.2^p + 1.5^p \cdot 24 + 1.51^p + 1.6^p \cdot 26$. Otherwise the cost is $1.5^p \cdot 48 + 1.51^p + 1.6^p \cdot 53 + (1.6 + 1.2 - 0.1\alpha^*)^p$. Subtracting off like terms, we conclude that the first case is always smaller because $1.51^p \cdot 49 + 1.1^p + 1.2^p < 1.5^p \cdot 24 + 1.6^p \cdot 26 + (1.6 + 1.2 - 0.1\alpha^*)^p$ for all $p \geq 1$.

Next, we will construct the second gadget arbitrarily far away from the first gadget. The second gadget is very similar to the first. There are points $a_2, b_2, c_2, x_2, y_2, x_2^*, y_2^*$ and sets to A_2, B_2, X_2, Y_2. $d(a_2, b_2) = d(x_2, y_2) = 1$, $d(x_2, c_2) = 1.1$, $d(y_2, c_2) = 1.2$, and for $b = 1$, $d(a_2, c_2) = d(b_2, c_2) = 1.2 - .1 \cdot \alpha^*$. For $b = 2$ or $b = 3$, $d(a_2, c_2) = d(b_2, c_2) = ((1.1^{\alpha^*} + 1.2^{\alpha^*})/2)^{\frac{1}{\alpha^*}}$. The rest of the distances are the same as in gadget 1. Then c_2 joins $\{a_2, b_2\}$ if $\alpha \geq \alpha^*$, not $\alpha \leq \alpha^*$. The rest of the argument is identical. So the conclusion we reach, is that the cost for the second gadget is much lower if $\alpha \geq \alpha^*$.

Therefore, the final cost of the 4-clustering is minimized when $\alpha = \alpha^*$, and the proof is complete.

□

3.3.2 Discretization

Another natural question to ask is whether a discretized set of the parameter space will always contain some parameter that is approximately optimal (for instance, an ϵ-net of the parameter space). In Corollary 3.3.3, we show this is not possible: for any data-independent discretization $D = \{d_1, \ldots, d_m\}$ of the parameter space, there exists an infinite family of clustering instances such that all $\alpha \in D$ will output a clustering that is an $\Omega(n)$ factor worse than the optimal value of α. First we prove the main structural idea behind Corollary 3.3.3.

Theorem 3.3.2. *For $b \in \{1, 2, 3\}$, for all $\frac{1}{3} < x < y < \frac{2}{3}$, $n > 10$, and $p \in O(1)$, there exists a clustering instance $\mathcal{V}$ such that for all $\alpha \in [x, y]$, $\texttt{clus}^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V}) \in O(1)$, and for all $\alpha \notin [x, y]$, $\texttt{clus}^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V}) \in \Omega(n)$.*

Proof. Given $\frac{1}{3} < x < y < \frac{2}{3}$ and $n > 10$, we will construct an instance $\mathcal{V}$ with the desired properties. We set $k = 2$.

Here is a high level description of our construction $\mathcal{V} = (V, d)$. There will be two gadgets. Gadget 1 contains points x_1, y_1, x_1', y_1', and z_1. Gadget 2 contains points x_2, y_2, x_2', y_2', and z_2. We will define the distances so the following merges take place. Initially, x_1 merges to y_1, x_1' merges to y_1', x_2 merges to y_2, and x_2' merges to y_2'. Then the sets are $\{x_1, y_1\}$, $\{x_1', y_1'\}$, $\{z_1\}$, $\{x_2, y_2\}$, $\{x_2', y_2'\}$, and $\{z_2\}$. Next, z_1 will merge to $\{x_1, y_1\}$ if $\alpha < x$, otherwise it will merge to $\{x_1', y_1'\}$. Similarly, z_2 will merge to $\{x_2, y_2\}$ if $\alpha < y$, otherwise it will merge to $\{x_2', y_2'\}$. Finally, the

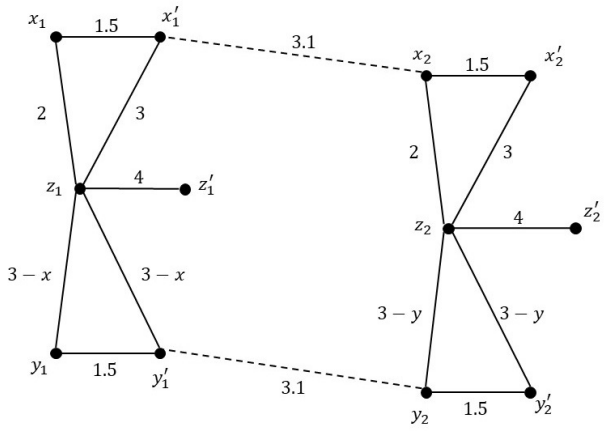


Figure 3.2: The clustering instance used in Theorem 3.3.2

sets containing $\{x_1, y_1\}$ and $\{x_2, y_2\}$ will merge, and the sets containing $\{x_1', y_1'\}$ and $\{x_2', y_2'\}$ will merge.

Therefore, the situation is as follows. If $\alpha \in [x, y]$, then the last two sets in the merge tree will each contain exactly one of the points $\{z_1, z_2\}$. If $\alpha \notin [x, y]$, then if we again look at the last two sets in the merge tree, one of the sets will contain both points $\{z_1, z_2\}$. Since these are the last two sets in the merge tree, the pruning step is not able to output a clustering with z_1 and z_2 in different clusters. To finish the proof, we give a high weight to points z_1 and z_2 by placing $\frac{n-8}{2}$ points in the same location as z_1, and $\frac{n-8}{2}$ points in the same location as z_2. Note this does not affect the merge equations. When z_1 and z_2 are in different clusters, the optimal centers for $k = 2$ are at z_1 and z_2, and the cost is just the cost of the remaining points, $\{x_1, x_1', y_1, y_1', x_2, x_2', y_2, y_2'\}$, and all distances will be between 1 and 6, so the total cost is $\leq 8 \cdot 6^p$. When z_1 and z_2 are in the same cluster, the center will be distance at least 2 from either z_1 or z_2 (or both), so the cost is $\geq \frac{n-8}{2} \cdot 2^p \in \Omega(n)$.

Now we define the distances and prove the desired merges take place in the correct ranges of α (see Figure 3.2). First we consider $\mathcal{A}_3$. We set

$$\begin{aligned} d(x_1, y_1) = d(x_2, y_2) = d(x_1', y_1') = d(x_2', y_2') &= 1.5, \\ d(x_1, z_1) = d(x_2, z_2) &= 2.4, \\ d(x_1', z_1) = d(x_2', z_2) &= 2.6, \\ d(y_1, z_1) = d(y_1', z_1) &= 2.6 - .2x, \\ d(y_2, z_2) = d(y_2', z_2) &= 2.6 - .2y, \end{aligned}$$

We set all distances between $\{x_1, y_1\}$ and $\{x_2, y_2\}$ to 2.7. Similarly, we set all distances between $\{x_1', y_1'\}$ and $\{x_2', y_2'\}$ to 2.7. All other distances are the maximum allowed by the triangle inequality.

Then, the first four merges are $\{x_1, y_1\}$, $\{x_1', y_1'\}$, $\{x_2, y_2\}$, and $\{x_2', y_2'\}$ since these are the shortest distances. The next-shortest distances are between 2.4 and 2.6, so z_1 will merge to either $\{x_1, y_1\}$ or $\{x_1', y_1'\}$, and z_2 will merge to either $\{x_2, y_2\}$ or $\{x_2', y_2'\}$. The decision for z_1 corresponds to the equation $\alpha \cdot 2.4 + (1 - \alpha) \cdot 2.6 = \alpha \cdot (2.6 - .2x) + (1 - \alpha) \cdot (2.6 - .2x)$, so z_1

will merge to $\{x_1, y_1\}$ if $\alpha < x$, otherwise it will merge to $\{x'_1, y'_1\}$. Similarly, we conclude that z_2 merges to $\{x_2, y_2\}$ if $\alpha < y$, otherwise $\{x'_2, y'_2\}$.

Next, we want the set containing $\{x_1, y_1\}$ to merge to the set containing $\{x_2, y_2\}$ and the set containing $\{x'_1, y'_1\}$ to merge to the set containing $\{x'_2, y'_2\}$. For both of these merges, the merge equation is $\alpha \cdot 2.7 + (1-\alpha)2.7 = 2.7$. However, the merge equation for $\{x_1, y_1\}$ to $\{x'_1, y'_1\}$ could be as small as $\alpha \cdot 2.4 + (1-\alpha) \cdot 4.8$, which is smaller than 2.7 for $\alpha > .875$. In order to ensure that this clustering instance has high cost when $\alpha > .875$, we add a few more points close to z_1 and z_2 which will cause a cluster containing z_1 and z_2 to merge early on, whenever $\alpha > .86$. Specifically, we set $d(z_1, z_2) = 2.4$ and add z'_1 and z'_2 such that $d(z_1, z'_1) = d(z_2, z'_2) = 1.1$, and the distances to $x_1, y_1, x'_1, y'_1, x_2, y_2, x'_2, y'_2$ are the same as for z_1 and z_2. So z_1 merges to z'_1 and z_2 merges to z'_2, and the merge equation for $\{z_1, z'_1\}$ and $\{z_2, z'_2\}$ is smaller than 2.7 when $\alpha > .86$. This will ensure the clustering has high cost when $\alpha > .86$.

Now, if $\alpha \in [x, y]$, then the last two sets in the merge tree are

$$\{x_1, y_1, x_2, y_2, z_1\} \text{ and } \{x'_1, y'_1, x'_2, y'_2, z_2\},$$

which each contain exactly one of the points $\{z_1, z_2\}$. If $\alpha \notin [x, y]$, then if we again look at the last two sets in the merge tree, one of the sets will contain both points $\{z_1, z_2\}$. Since these are the last two sets in the merge tree, the pruning step is not able to output a clustering with z_1 and z_2 in different clusters. When z_1 and z_2 are in different clusters, since they both have high weight, the optimal centers for $k = 2$ are at z_1 and z_2, and the cost of the remaining 8 points is at most $8 \cdot 6^p$. When z_1 and z_2 are in the same cluster, the center is distance 2 from at least one of them, so the cost is $\geq \frac{n-8}{2} \cdot 2^p$. When p is a constant, the difference in cost between these cases is $\Omega(n)$.

The cases for $\mathcal{A}_1$ and $\mathcal{A}_2$ are similar to the previous case. All distances are the same, except we set

$$d(y_1, z_1) = d(y'_1, z_1) = \left(\frac{1}{2}\left(2.4^x + 2.6^x\right)\right)^{\frac{1}{x}},$$
$$d(y_2, z_2) = d(y'_2, z_2) = \left(\frac{1}{2}\left(2.4^y + 2.6^y\right)\right)^{\frac{1}{y}}.$$

This ensures that z_1 will merge to $\{x_1, y_1\}$ if $\alpha < x$, otherwise it will merge to $\{x'_1, y'_1\}$, and z_2 will merge to $\{x_2, y_2\}$ if $\alpha < y$, otherwise it will merge to $\{x'_2, y'_2\}$. The rest of the details of the proof are identical to the previous case. This concludes the proof. □

Now we can prove Corollary 3.3.3.

Corollary 3.3.3. *For $b \in \{1, 2, 3\}$ and $p \in O(1)$, given a finite discretization $D = \{d_1, \ldots, d_m\}$ of the parameter space, there exists a constant c such that for all $n > 10$, there exists a clustering instance $\mathcal{V}$ of size n such that $c \cdot n \cdot \min_{\alpha \in [0,1]} \texttt{clus}^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V}) < \min_{\alpha \in D} \texttt{clus}^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V})$.*

Proof. Given a discretization $D = \{d_1, \ldots, d_m\}$, note that $[0 = d_0, d_1], [d_1, d_2], \ldots, [d_m, d_{m+1} = 1]$ is a partition of the parameter space $[0, 1]$. Choose an interval $[d_i, d_{i+1}]$ which has nonempty intersection with $(\frac{1}{3}, \frac{2}{3})$. Now define a new interval $[d'_i, d'_{i+1}]$ such that $d'_i = \max(d_i, \frac{1}{3})$ and $d'_{i+1} =$

$\min(d_{i+1}, \frac{2}{3})$. We set $x = d'_i + \frac{d'_{i+1} - d'_i}{3}$ and $y = d'_{i+1} - \frac{d'_{i+1} - d'_i}{3}$. By construction, we have $[x, y] \subseteq (d_i, d_{i+1})$ and $[x, y] \subseteq (\frac{1}{3}, \frac{2}{3})$, and it follows that $D \cap [x, y] = \emptyset$. Now for each $n > 10$, we use Theorem 3.3.2 with x and y as defined above, to obtain $\mathcal{V}$ such that for all $\alpha \in [x, y]$, $\text{cost}(\mathcal{V}, \alpha) \in O(1)$, and for all $\alpha \notin [x, y]$ (including all of D), $\text{cost}(\mathcal{V}, \alpha) \in \Omega(n)$. This completes the proof. □

3.3.3 Pseudo-dimension upper bounds

Now for an arbitrary objective function Φ and arbitrary pruning function Ψ, we analyze the complexity of the classes

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

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 $\texttt{clus}_{\mathcal{A}_i, \mathcal{V}}(\alpha)$ to analyze how $\texttt{clus}_{\mathcal{A}_i(\alpha)}(\mathcal{V})$ changes as a function of α. We start with $\mathcal{A}_1$ and $\mathcal{A}_3$.

Theorem 3.3.4. *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$\left(\mathcal{H}_{\mathcal{A}_3 \times \{\Psi\}, \Phi}\right) = O(\log n)$.*

This theorem follows from Lemma 3.3.8 and Lemma 3.3.13. We begin with the following structural lemma, which will help us prove Lemma 3.3.8.

Lemma 3.3.5. *$\texttt{clus}_{\mathcal{A}_3, \mathcal{V}} : [0, 1] \to \mathbb{R}_{>0}$ is made up of $O(n^8)$ piecewise constant components.*

Proof. First 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. Now, as we increase α from 0 to 1 and observe the run of the algorithm for each α, at what values of α do we expect $\mathcal{A}_1(\alpha)$ to produce different merge trees? To answer this, suppose that at some point in the run of algorithm $\mathcal{A}_1(\alpha)$, there are two 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 whether $\alpha d(p, q) + (1 - \alpha) d(p', q')$ or $\alpha d(x, y) + (1 - \alpha) d(x', y')$ is larger. This is a linear equation in α, so there is at most one value of α for which these expressions are equal, unless the difference of the expressions is zero for all α. Assuming that ties are broken arbitrarily but consistently, this implies that there is at most one $\alpha \in [0, 1]$ such that the choice of whether to merge (A, B) before (X, Y) is identical for all $\alpha < \alpha'$, and similarly identical for $\alpha \geq \alpha'$. Since each merge decision is defined by eight points, iterating over all pairs (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 $[0, 1]$ into $O(n^8)$ intervals over each of which the merge tree, and therefore the output of $\texttt{clus}_{\mathcal{A}_1, \mathcal{V}}(\alpha)$, is fixed. □

[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.

Now we will provide the details of Lemma 3.3.7. In the argument for the structure of $\mathcal{H}_{\mathcal{A}_3,\texttt{clus}}$, we relied on the linearity of $\mathcal{A}_3$'s merge equation to prove that for any eight points, there is exactly one value of α such that $\alpha d(p,q) + (1-\alpha)d(p',q') = \alpha d(x,y) + (1-\alpha)d(x',y')$. Now we will use Theorem 3.3.6, a consequence of Rolle's Theorem, to bound the values of α such that $((d(p,q))^\alpha + d(p',q')^\alpha)^{1/\alpha} = ((d(x,y))^\alpha + d(x',y')^\alpha)^{1/\alpha}$.

Theorem 3.3.6 (ex. Tossavainen [2006])**.** *Let f be a polynomial-exponential sum of the form $f(x) = \sum_{i=1}^{N} a_i b_i^x$, where $b_i > 0$, $a_i \in \mathbb{R}$, and at least one a_i is non-zero. The number of roots of f is upper bounded by N.*

Theorem 3.3.7. $\texttt{clus}_{\mathcal{A}_1,\mathcal{V}} : \mathbb{R} \cup \{-\infty, \infty\} \to \mathbb{R}_{>0}$ *is made up of $O(n^8)$ piecewise constant components.*

Proof. As was the case for $\mathcal{H}_{\mathcal{A}_3}$, the clustering returned by $\mathcal{A}_1(\alpha)$ and the associated cost are identical to that of $\mathcal{A}_1(\alpha')$ as long as both algorithms construct the same merge trees. Our objective is to understand the behavior of $\mathcal{A}_1(\alpha)$ over m instances. In particular, as α varies over $\mathbb{R}$ we want to count the number of times the algorithm outputs a different merge tree on one of these instances. For some instance $\mathcal{V}$ we will consider two pairs of sets A, B and X, Y that can be potentially merged. The decision to merge one pair before the other is determined by the sign of $d^\alpha(p,q) + d^\alpha(p',q') - d^\alpha(x,y) + d^\alpha(x',y')$. This expression, as before, is determined by a set of 8 points $p, p' \in A$, $q, q' \in B$, $x, x' \in X$ and $y, y' \in Y$ chosen independent of α.

Now, from Theorem 3.3.6, we have that the sign of the above expression as a function of α flips at most 4 times across $\mathbb{R}$. Since the expression is defined by exactly 8 points, iterating over all pairs (A, B) and (X, Y) we can list only $O(n^8)$ such unique expressions, each of which correspond to $O(1)$ values of α at which the corresponding decision flips. Thus, we can divide $\mathbb{R}$ into $O(n^8)$ intervals over each of which the output of $\texttt{clus}_{\mathcal{A}_1,\mathcal{V}}(\alpha)$ is fixed. □

These lemmas allow us to upper bound the pseudo-dimension of $\mathcal{H}_{\mathcal{A}_1}, \texttt{clus}$ and $\mathcal{H}_{\mathcal{A}_1}, \texttt{clus}$ by $O(\log n)$. Thus we obtain the following lemma.

Theorem 3.3.8. *For any* $\texttt{clus}$*, Pdim*$(\mathcal{H}_{\mathcal{A}_1}, \texttt{clus}) = O(\log n)$ *and Pdim*$(\mathcal{H}_{\mathcal{A}_3}, \texttt{clus}) = O(\log n)$.

Proof. Suppose $\mathcal{S} = \left\{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}\right\}$ is a set of clustering instances that can be shattered by $\mathcal{H}_{\mathcal{A}_1}$ using the witnesses $r_1, \ldots, r_m$. We must show that $m = O(\log n)$. For each value of $\alpha \in \mathbb{R} \cup \{-\infty, \infty\}$, the algorithm $\mathcal{A}_1(\alpha)$ induces a binary labeling on each $\mathcal{V}^{(i)}$, based on whether or not $\texttt{clus}_{\mathcal{A}_1(\alpha)}\left(\mathcal{V}^{(i)}\right) \leq r_i$. From Lemma 3.3.7, we know that every sample $\mathcal{V}^{(i)}$ partitions $\mathbb{R} \cup \{\infty, -\infty\}$ into $O(n^8)$ intervals in this way. Merging all m partitions, we can divide $\mathbb{R} \cup \{\infty, -\infty\}$ into $O(mn^8)$ intervals over each of which $\texttt{clus}_{\mathcal{A}_3,\mathcal{V}^{(i)}}(\alpha)$, and therefore the labeling induced by the witnesses, is fixed for all $i \in [m]$. This means that $\mathcal{H}_{\mathcal{A}_1}$ can achieve only $O(mn^8)$ binary labelings, which is at least 2^m since $\mathcal{S}$ is shatterable, so $m = O(\log n)$.

The details for $\mathcal{H}_{\mathcal{A}_3}, \texttt{clus}$ are identical, by using Lemma 3.3.5. □

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

Theorem 3.3.9. *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*$\left(\mathcal{H}_{\mathcal{A}_2\times\{\Psi\},\Phi}\right) = O(n)$.

This theorem follows from Lemmas 3.3.10 and 3.3.15.

Lemma 3.3.10. *For all objective functions Φ and all pruning functions Ψ, Pdim$\left(\mathcal{H}_{\mathcal{A}_2\times\{\Psi\},\Phi}\right) = O(n)$.*

Proof. Recall the proof of Lemma 3.3.8. We are interested in studying how the merge trees constructed by $\mathcal{A}_2(\alpha)$ changes over m instances as we increase α over $\mathbb{R}$. To do this, as in the proof of Lemma 3.3.8, we fix an instance and consider two pairs of sets A, B and X, Y that could be potentially merged. Now, the decision to merge one pair before the other is determined by the sign of the expression $\frac{1}{|A||B|}\sum_{p\in A,q\in B}(d(p,q))^\alpha - \frac{1}{|X||Y|}\sum_{x\in X,y\in Y}(d(x,y))^\alpha$. First note that this expression has $O(n^2)$ terms, and by Theorem 3.3.6, it has $O(n^2)$ roots. Therefore, as we iterate over the $O\left((3^n)^2\right)$ possible pairs (A, B) and (X, Y), we can determine $O\left(3^{2n}\right)$ unique expressions each with $O(n^2)$ values of α at which the corresponding decision flips. Thus we can divide $\mathbb{R}$ into $O\left(n^2 3^{2n}\right)$ intervals over each of which the output of $\Phi_{\mathcal{A}_3,\mathcal{V}}(\alpha)$ is fixed. In fact, suppose $\mathcal{S} = \left\{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}\right\}$ is a shatterable set of size m with witnesses $r_1, \ldots, r_m$. We can divide $\mathbb{R}$ into $O\left(mn^2 3^{2n}\right)$ intervals over each of which $\Phi_{\mathcal{A}_2,\mathcal{V}^{(i)}}(\alpha)$ is fixed for all $i \in [m]$ and therefore the corresponding labeling of $\mathcal{S}$ according to whether or not $\Phi_{\mathcal{A}_2(\alpha)}\left(\mathcal{V}^{(i)}\right) \le r_i$ is fixed as well for all $i \in [m]$. This means that $\mathcal{H}_{\mathcal{A}_2}$ can achieve only $O\left(mn^2 3^{2n}\right)$ labelings, which is at least 2^m for a shatterable set $\mathcal{S}$, so $m = O(n)$. □

3.3.4 Efficient algorithms

The upper bound on the pseudo-dimension implies a computationally efficient and sample efficient learning algorithm for $\mathcal{A}_i$ for $i \in \{1, 3\}$. See Algorithm 7. First, we know that $m = \tilde{O}\left((H/\epsilon)^2\right)$ samples are sufficient to (ϵ, δ)-learn the optimal algorithm in $\mathcal{A}_i$. Next, as a consequence of Lemmas 3.3.7 and 3.3.5, 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 3.3.11. *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 $i \in \{1, 3\}$, Algorithm 7 (ϵ, δ)-learns the class $\mathcal{A}_i \times \{\Psi\}$ with respect to the cost function Φ and it is computationally efficient.*

Proof. Algorithm 7 finds the best α over the sample by solving for the $O(mn^8)$ discontinuities of the function $\sum_{\mathcal{V}\in\mathcal{S}} \Phi_{\mathcal{A}_b(\alpha)}(\mathcal{V})$ and evaluating the function over the corresponding intervals, which are guaranteed to be constant by Lemmas 3.3.7 and 3.3.5. Therefore, we can pick any arbitrary α within each interval to evaluate the empirical cost over all samples, and find the empirically best α. This can be done in polynomial time because there are polynomially many intervals, and the runtime of $\mathcal{A}_i(\alpha)$ on a given instance is polynomial time.

Then it follows from Theorem 3.3.4 that m samples are sufficient for Algorithm 7 to (ϵ, δ)-learn the optimal algorithm in $\mathcal{A}_i$ for $i \in \{1, 3\}$. □

Algorithm 7 An algorithm for finding an empirical cost minimizing algorithm in $\mathcal{A}_1$ or $\mathcal{A}_3$

Input: Sample $\mathcal{S} = \left\{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}\right\}$, $b \in \{1, 3\}$

1: Let $T = \emptyset$. For each sample $\mathcal{V}^{(i)} = \left(V^{(i)}, d^{(i)}\right) \in \mathcal{S}$, and for each ordered set of 8 points $\{v_1, \ldots, v_8\} \subseteq V^{(i)}$, solve for α (if a solution exists) in the following equation and add the solutions to T: $d(v_1, v_2)^\alpha + d(v_3, v_4)^\alpha = d(v_5, v_6)^\alpha + d(v_7, v_8)^\alpha$.

$$\begin{aligned} \text{If } b = 1: \quad & d(v_1, v_2)^\alpha + d(v_3, v_4)^\alpha &=& \quad d(v_5, v_6)^\alpha + d(v_7, v_8)^\alpha. \\ \text{If } b = 3: \quad & \alpha d(v_1, v_2) + (1-\alpha) d(v_3, v_4) &=& \quad \alpha d(v_5, v_6) + (1-\alpha) d(v_7, v_8). \end{aligned}$$

2: Order the elements of set $T \cup \{-\infty, +\infty\}$ as $\alpha_1 < \ldots < \alpha_{|T|}$. For each $0 \leq i \leq |T|$, pick an arbitrary α in the interval (α_i, α_{i+1}) and run $\mathcal{A}_b(\alpha)$ on all clustering instances in $\mathcal{S}$ to compute $\sum_{\mathcal{V} \in \mathcal{S}} \Phi_{\mathcal{A}_b(\alpha)}(\mathcal{V})$. Let $\hat{\alpha}$ be the value which minimizes $\sum_{\mathcal{V} \in \mathcal{S}} \Phi_{\mathcal{A}_b(\alpha)}(\mathcal{V})$.

Output: $\hat{\alpha}$

Algorithm 8 An algorithm for finding an empirical cost minimizing algorithm in $\mathcal{A}_2$

Input: Sample $\mathcal{S} = \left\{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}\right\}$.

1: Let $T = \emptyset$. For each sample $\mathcal{V}^{(i)} = \left(V^{(i)}, d^{(i)}\right) \in \mathcal{S}$, and for all $A, B, X, Y \subseteq V^{(i)}$, solve for α (if a solution exists) in the following equation and add the solutions to T:

$$\frac{1}{|A||B|} \sum_{p \in A, q \in B} (d(p, q))^\alpha = \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} (d(x, y))^\alpha.$$

2: Order the elements of set $T \cup \{-\infty, +\infty\}$ as $\alpha_1 < \ldots < \alpha_{|T|}$. For each $0 \leq i \leq |T|$, pick an arbitrary α in the interval (α_i, α_{i+1}) and run $\mathcal{A}_2(\alpha)$ on all clustering instances in $\mathcal{S}$ to compute $\sum_{\mathcal{V} \in \mathcal{S}} \Phi_{\mathcal{A}_2(\alpha)}(\mathcal{V})$. Let $\hat{\alpha}$ be the value which minimizes $\sum_{\mathcal{V} \in \mathcal{S}} \Phi_{\mathcal{A}_2(\alpha)}(\mathcal{V})$.

Output: $\hat{\alpha}$

Now we give an ERM algorithm for $\mathcal{A}_2$, similar to Algorithm 7.

Theorem 3.3.12. *Let Φ be a clustering objective and let Ψ be a pruning function. Given an input sample of size $m = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(n + \log \frac{1}{\delta}\right)\right)$, Algorithm 8 (ϵ, δ)-learns the class $\mathcal{A}_2 \times \{\Psi\}$ with respect to the cost function Φ.*

Proof. The sample complexity analysis follows the same logic as the proof of Theorem 3.3.11. To prove that Algorithm 8 indeed finds the empirically best α, recall from the pseudo-dimension analysis that the cost as a function of α for any instance is a piecewise constant function with $O(n^2 3^{2n})$ discontinuities. In Step 1 of Algorithm 8, we solve for the values of α at which the discontinuities occur and add them to the set T. T therefore partitions α's range into $O(mn^2 3^{2n})$ subintervals. Within each of these intervals, $\sum_{\mathcal{V} \in \mathcal{S}} \Phi_{\mathcal{A}_2(\alpha)}(\mathcal{V})$ is a constant function. Therefore, we pick any arbitrary α within each interval to evaluate the empirical cost over all samples, and find the empirically best α. □

3.3.5 Pseudo-dimension lower bounds

Next, we give lower bounds for the pseudo-dimension of the two classes.

Lemma 3.3.13. *For any objective function $\Phi^{(p)}$, Pdim$\left(\mathcal{H}_{\mathcal{A}_1,\Phi^{(p)}}\right) = \Omega(\log n)$ and Pdim$\left(\mathcal{H}_{\mathcal{A}_3,\Phi^{(p)}}\right) = \Omega(\log n)$.*

First we give a general proof outline for both classes. Let $b \in \{1, 3\}$. We construct a set $S = \left\{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}\right\}$ of $m = \log n - 3$ clustering instances that can be shattered by $\mathcal{A}_b$. There are $2^m = n/8$ possible labelings for this set, so we need to show there are $n/8$ choices of α such that each of these labelings is achievable by some $\mathcal{A}_b(\alpha)$ for some α. The crux of the proof lies in showing that given a sequence $\alpha_0 < \alpha_1 < \cdots < \alpha_{n'} < \alpha_{n'+1}$ (where $n' = \Omega(n)$), it is possible to design an instance $\mathcal{V} = (V, d)$ over n points and choose a witness r such that $\texttt{clus}_{\mathcal{A}_b(\alpha)}(\mathcal{V})$ alternates $n'/2$ times above and below r as α traverses the sequence of intervals (α_i, α_{i+1}).

Here is a high level description of our construction. There will be two "main" points, a and a' in V. The rest of the points are defined in groups of 6: $(x_i, y_i, z_i, x_i', y_i', z_i')$, for $1 \le i \le (n-2)/6$. We will define the distances between all points such that initially for all $\mathcal{A}_b(\alpha)$, x_i merges to y_i to form the set A_i, and x_i' merges to y_i' to form the set A_i'. As for (z_i, z_i'), depending on whether $\alpha < \alpha_i$ or not, $\mathcal{A}_b(\alpha)$ merges the points z_i and z_i' with the sets A_i and A_i' respectively or vice versa. This means that there are $(n-2)/6$ values of α such that $\mathcal{A}_b(\alpha)$ has a unique behavior in the merge step. Finally, for all α, sets A_i merge to $\{a\}$, and sets A_i' merge to $\{a'\}$. Let $A = \{a\} \cup \bigcup_i A_i$ and $A' = \{a'\} \cup \bigcup_i A_i'$. There will be $(n-2)/6$ intervals (α_i, α_{i+1}) for which $\mathcal{A}_b(\alpha)$ returns a unique partition $\{A, A'\}$. By carefully setting the distances, we cause the cost $\Phi(\{A, A'\})$ to oscillate above and below a specified value r along these intervals.

Now we give the full details of the proof of Lemma 3.3.13 We first prove this lemma for the center-based objective cost denoted by $\Phi^{(p)}$ for $p \in [1, \infty) \cup \{\infty\}$. We later note how this can be extended cluster purity based cost. We first prove the following useful statement which helps us construct general examples with desirable properties. In particular, the following lemma guarantees that given a sequence of values of α of size $O(n)$, it is possible to construct an instance $\mathcal{V}$ such that the cost of the output of $\mathcal{A}_1(\alpha)$ on $\mathcal{V}$ as a function of α, that is $\Phi^{(p)}_{\mathcal{A}_1,\mathcal{V}}(\alpha)$, oscillates above and below some threshold as α moves along the sequence of intervals (α_i, α_{i+1}). Given this powerful guarantee, we can then pick appropriate sequences of α and generate a sample set of $\Omega(\log n)$ instances that correspond to cost functions that oscillate in a manner that helps us pick $\Omega(n)$ values of s that shatters the samples. We also show how to trade off the number of oscillations, with the difference in cost between the oscillations, using parameter γ. However, $\gamma = 1$ is sufficient to obtain a pseudo-dimension lower bound.

Lemma 3.3.14. *Given $n \in \mathbb{N}$, $0 < \gamma \le 1$, $b \in \{1, 3\}$, $\gamma \le 1$, and given a sequence of $n' \le \lfloor \frac{\gamma n}{7} \rfloor$ α's such that $.3 = \alpha_0 < \alpha_1 < \cdots < \alpha_{n'} < \alpha_{n'+1} = .6$, there exists a real valued witness $r > 0$ and a clustering instance $\mathcal{V} = (V, d)$, $|V| = n$, such that for $0 \le i \le n'/2 - 1$, $\Phi^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V}) < \gamma \cdot r$ for $\alpha \in (\alpha_{2i}, \alpha_{2i+1})$, and $\Phi^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V}) > r$ for $\alpha \in (\alpha_{2i+1}, \alpha_{2i+2})$, for $k = 2$.*

Proof. The idea of the proof is as follows. There will be two "main" points, a and a' in V. The rest of the points are defined in groups of 6: $(x_i, y_i, z_i, x_i', y_i', z_i')$, for $1 \le i \le (n-2)/6$. We will define the distances between all points such that initially for all $\mathcal{A}_b(\alpha)$, x_i merges to y_i to form the

set A_i, and x_i' merges to y_i' to form the set A_i'. As for z_i and z_i', depending on whether $\alpha < \alpha_i$ or not, $\mathcal{A}_b(\alpha)$ merges the points z_i and z_i' with the sets A_i and A_i' respectively or vice versa. This means that there are $(n-2)/6$ values of α such that $\mathcal{A}_b(\alpha)$ has a unique behavior in the merge step. Finally, for all α, sets A_i merge to $\{a\}$, and sets A_i' merge to $\{a'\}$. Let $A = \{a\} \cup \bigcup_i A_i$ and $A' = \{a'\} \cup \bigcup_i A_i'$. There will be $(n-2)/6$ intervals (α_i, α_{i+1}) for which $\mathcal{A}_b(\alpha)$ returns a unique partition $\{A, A'\}$. By carefully setting the distances, we cause the cost $\Phi(\{A, A'\})$ to oscillate above and below a specified value r along these intervals. In order to make the cost oscillate above r and below γr, we give a high weight to two points, by putting $\frac{1-\gamma}{2}$ points in the same location as these two points. The first high-weight point is a, and the second high-weight point is a new point z. We set the distances so that z oscillates between merging to A or A' as we increase α from .3 to .6. If z merges to A', then the 2-clustering cost is low because we can put centers on a and z. If z merges to A, then both a and z are in the same cluster, incurring a large cost.

Now we will give the full details of the proof, including all distances. First we focus on $\mathcal{A}_3$ and $\gamma = 1$, and we discuss the other cases later in the proof. First of all, in order for d to be a metric, we set all distances in $[1, 2]$ so that the triangle inequality is trivially satisfied. The following are the distances of the pairs of points for $1 \leq i \leq (n-2)/6$.

$$\begin{aligned}
&d(x_i, y_i) = d(x_i', y_i') = 1,\\
&d(x_i, z_i) = 1.3,\ d(y_i, z_i) = 1.4,\\
&d(x_i', z_i) = d(y_i', z_i) = 1.4 - .1 \cdot \alpha_i,\\
&d(x_i, x_i') = d(y_i, y_i') = 2.
\end{aligned}$$

We set the distances to z_i' as follows (see Figure 3.3).

$$\begin{aligned}
&d(x_i, z_i') = d(y_i, z_i') = d(x_i', z_i') = d(y_i', z_i') = 1.41,\\
&d(z_i, z_i') = 2.
\end{aligned}$$

Then the first merges will be x_i to y_i and x_i' to y_i', no matter what α is set to be (when each point is a singleton set, each pair of points with the minimum distance in the metric will merge). Next, z_i will either merge to A_i or A_i' based on the following equation:

$$\begin{aligned}
& \alpha \cdot 1.3 + (1-\alpha) \cdot 1.4 \lesseqgtr \alpha \cdot (1.4 - .1 \cdot \alpha_i) + (1-\alpha)(1.4 - .1 \cdot \alpha_i)\\
\implies \quad & 1.4 - .1 \cdot \alpha \lesseqgtr 1.4 - .1 \cdot \alpha_i\\
\implies \quad & \alpha_i \lesseqgtr \alpha
\end{aligned}$$

If $\alpha < \alpha_i$, then z_i merges to A_i', otherwise it will merge to A_i. After one of these merges takes place, the new value for merging A_i to A_i' could be as small as $\alpha \cdot 1.3 + (1-\alpha) \cdot 2 = 2 - .6 \cdot \alpha$, but we do not want this merge to occur. If we ensure all subsequent merges have maximum distance less than 1.5, then A_i will not merge to A_i' (until A and A' merge in the very final step) as long as $\alpha < .6$, because $\alpha \cdot 1.5 + (1-\alpha) \cdot 1.5 = 1.5 < 2 - .6 \cdot .7$.

These distances ensure z_i' merges after z_i regardless of the value of α, since z_i is closer than z_i' to x_i, x_i', y_i, and y_i'. Furthermore, z_i' will merge to the opposite set of z_i, since we set $d(z_i, z_i') = 2$.

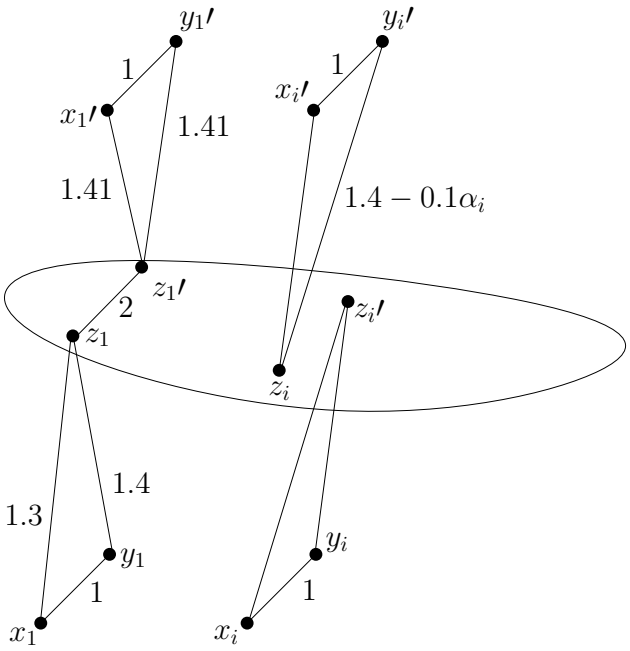


Figure 3.3: The clustering instance used in Lemma 3.3.14

The merge expression for z_i' to merge to the opposite set is $\alpha \cdot 1.41 + (1-\alpha) \cdot 1.41$, while the merge expression to the same set is $\geq \alpha \cdot 1.41 + (1-\alpha) \cdot 2$.

Now we set the distances to a and a' as follows.

$$d(a, x_i) = d(a, y_i) = d(a', x_i') = d(a', y_i') = 1.42,$$
$$d(a, x_i') = d(a, y_i') = d(a', x_i) = d(a', y_i') = 2.$$

We also set all distances between A_i and A_j' to be 2, for all i and j, and all distances between A_i and A_j to be 1.5, for all $i \neq j$. We will set the distances from a and a' to z_i and z_i' later, but they will all fall between 1.46 and 1.47. By construction, every set A_i will merge to the current superset containing $\{a\}$, because the merge expression is $\alpha \cdot 1.42 + (1-\alpha)1.5$, and *any other possible merge* will have value $\geq \alpha \cdot 1.3 + (1-\alpha) \cdot 2$, which is larger for $\alpha < .6$. Similarly, all A_i' sets will merge to $\{a'\}$.

Therefore, the final two sets in the linkage tree are A and A'. Given $1 \leq i \leq (n-2)/6$, by construction, for $\alpha \in (\alpha_i, \alpha_{i+1})$,

$$\{z_1, \ldots, z_i, z_{i+1}', \ldots z_{(n-2)/6}'\} \subseteq A \text{ and } \{z_1', \ldots, z_i', z_{i+1}, \ldots z_{(n-2)/6}\} \subseteq A'.$$

Finally, we set the distances between a, a', z_i, and z_i' to ensure the cost function oscillates.

$$\begin{aligned} \forall i, \quad & d(a, z_i') = d(a', z_i) = 1.46 \\ \forall 1 \leq j \leq (n-2)/12, \quad & d(a, z_{2j-1}) = d(a', z_{2j}') = 1.47, \\ & \text{and } d(a, z_{2j}) = d(a', z_{2j+1}') = (2 \cdot 1.46^p - 1.47^p)^{1/p}. \end{aligned}$$

Now we calculate the 2-clustering cost of (A, A') for α's in different ranges. Regardless of α, all partitions will pay $\sum_i (d(a, x_i)^p + d(a, y_i)^p + d(a', x_i')^p + d(a', y_i')^p) = (n-2)/6 \cdot (4 \cdot 1.42^p)$,but

the distances for z_i and z'_i differ. For $\alpha \in (\alpha_0, \alpha_1)$, all of the z's pay 1.46^p, so the cost is $(n-2)/6 \cdot (4 \cdot 1.42^p + 2 \cdot 1.46^p)$. Denote this value by r_{low}.

When $\alpha \in (\alpha_1, \alpha_2)$, the only values that change are z_1 and z'_1, which adds $d(a, z_1) + d(a', z'_1) - d(a, z'_1) - d(a', z_1) = 2 \cdot (1.47^p - 1.46^p) > 0$ to the cost (the inequality is always true for $p \in [1, \infty]$). Denote $r_{low} + 2 \cdot (1.47^p - 1.46^p)$ by r_{high}. When $\alpha \in (\alpha_2, \alpha_3)$, the values of z_2 and z'_2 change, and the cost changes by $d(a, z_2) + d(a', z'_2) - d(a, z'_2) - d(a', z_2) = 2 \cdot ((2 \cdot 1.46^p - 1.47^p) - 1.46^p) = -2 \cdot (1.47^p - 1.46^p)$, decreasing it back to r_{low}.

In general, the cost for $\alpha \in (\alpha_i, \alpha_{i+1})$ is $r_{low} + \sum_{1 \le j \le i} (-1)^{i+1} \cdot 2(1.47^p - 1.46^p) = r_{low} + (1.47^p - 1.46^p) + (-1)^{i+1} \cdot (1.47^p - 1.46^p)$. If $\alpha \in (\alpha_{2j}, \alpha_{2j+1})$, then the cost is r_{low}, and if $\alpha \in (\alpha_{2j+1}, \alpha_{2j+2})$, the cost is r_{high}. We set $r = (r_{low} + r_{high})/2$, and conclude that the cost function oscillates above and below r as specified in the lemma statement.

The pruning step will clearly pick (A, A') as the optimal clustering, since the only centers with more than 3 points at distance < 1.5 are a and a', and (A, A') are the clusters in which the most points can have a and a' as centers. This argument proved the case where $n' = (n-2)/6$. If $n' < (n-2)/6$, then we set $d(a, z_i) = d(a', z'_i) = 1.46$ for all $i > n'$, which ensures the cost function oscillates exactly n' times. This completes the proof for $\mathcal{A}_3$ and $\gamma = 1$.

It is straightforward to modify this proof to work for $\mathcal{A}_1$. The only major change is to set

$$d(x'_i, z_i) = d(y'_i, z_i) = ((1.3_i^\alpha + 1.4_i^\alpha)/2)^{\frac{1}{\alpha_i}}.$$

Now we move to the case where $\gamma < 1$. In this case, the cost will oscillate between $> r$ and $< \gamma \cdot r$, for a value of r defined later. To accomplish this, we put large weight on a and a new point z. Our goal is to show that the optimal $k = 2$ pruning oscillates between putting a and z in the same cluster, versus different clusters, for the intervals defined by $\alpha_0, \ldots, \alpha_{n'}$. We will use $\frac{\gamma \cdot n}{7}$ gadgets consisting of 6 points each, to achieve $\frac{\gamma \cdot n}{7}$ intervals that oscillate. The remaining $(1-\gamma)n$ points will be used to create a separation between the costs of the optimal 2-clustering in neighboring α intervals.

Now we will show how to alternate a and z in the same cluster versus different clusters. Note that a will always merge to the set A by definition. Next we set the distances from $z_1, \ldots, z_{n'}$ and $z'_1, \ldots, z'_{n'}$ to z so that it alternates merging to A or A' along $\alpha_0, \ldots, \alpha_{n'}$. We set the distances between z and $z_1, \ldots, z_{n'}, z'_1, \ldots, z'_{n'}$ as follows. We nest $d(z, z_1) < \cdots < d(z, z_{n'}) < d(z, z_{n'}) < d(z, z'_1)$. Recall that in interval (α_i, α_{i+1}), A contains $z_1, \ldots, z_i, z'_{i+1}, \ldots, z'_{n'}$, and A' contains $z'_1, \ldots, z'_i, z'_{i+1}, \ldots, z'_{n'}$. Therefore, the merge equation in this interval is

$$\alpha_i d(z, z_1) + (1 - \alpha_i) d(z, z'_1) \lesseqgtr \alpha_i d(z, z_{i+1}) + (1 - \alpha_i) d(z, z'_{i+1}).$$

We set $d(z, z_1) = 1.46$, $d(z, z'_1) = 1.47$, and $d(z, z'_i) - d(z, z_i) = \frac{1}{2^i}$. Then we solve to find $d(z, z_i) = 1.47 - .01\alpha_1 + \frac{\alpha_i}{2^i}$ and $d(g_2, z'_i) = 1.47 - .01\alpha_1 - \frac{1}{2^i}(1 - \alpha_i)$ would achieve equality in the equation above. Call these values d_i and d'_i, respectively. If we set the distances to exactly these values, then we would have exact ties for the decision to merge z to A or A' in all α intervals. Therefore, we add small offsets of size $\epsilon = .0001$ to some of the values. Specifically, set $d(z, z_i) = d_i$ for all i. For even i, set $d(z, z'_i) = d'_i + \epsilon$, and for odd i, set $d(z, z'_i) = d'_i - \epsilon$. This ensures z oscillates merging to A or A' along the n' α intervals.

The pruning step for $k = 2$ must output A and A', since this is the last merge that takes place in the tree. When z is in A', then the optimal centers are at a and z, and the cost of the clustering is the cost of the γn points making up the gadgets, which is $O(\gamma n)$. When z is in A, then the center for A must be distance at least 1 to either a or z, so the cost of the clustering is at least $\frac{1-\gamma}{2} \cdot n$. Therefore, the difference in cost is $\Omega\left(\frac{1-\gamma}{\gamma}\right)$. □

Now we can prove Lemma 3.3.13.

Proof of Lemma 3.3.13. Given $b \in \{1, 3\}$, we prove the claim for $\mathcal{H}_{\mathcal{A}_b,\texttt{clus}^{(p)}}$ by constructing a set of samples $\mathcal{S} = \{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}\}$ where $m = \log n - 3$ that can be shattered by $\mathcal{H}_{\mathcal{A}_b,\texttt{clus}^{(p)}}$. That is, we should be able to choose $2^m = n/8$ different values of α such that there exists some witnesses $r_1, \ldots, r_m$ with respect to which $\Phi^{(p)}_{\mathcal{A}_b(\alpha)}(\cdot)$ induces all possible labelings on $\mathcal{S}$.

Choose a sequence of 2^m distinct α's arbitrarily in the range $(0, .7)$. We will index the terms of this sequence using the notation $\alpha_{\mathbf{x}}$ for all $\mathbf{x} \in \{0, 1\}^m$, such that $\alpha_{\mathbf{x}} < \alpha_{\mathbf{y}}$ iff $\mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_m < \mathbf{y}_1\mathbf{y}_2\mathbf{y}_m$. Then the α's satisfy

$$0 < \alpha_{[0\ \ldots\ 0\ 0]} < \alpha_{[0\ \ldots\ 0\ 1]} < \alpha_{[0\ \ldots\ 1\ 0]} < \cdots < \alpha_{[1\ \ldots\ 1\ 1]} < .7.$$

Given $\mathbf{x}$, denote by $n(\mathbf{x})$ the vector corresponding to $\mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_s + 1$, therefore, $\alpha_{n(\mathbf{x})}$ is the smallest α greater than $\alpha_{\mathbf{x}}$.

Now, the crucial step is that we will use Lemma 3.3.14 to define our examples $\mathcal{V}^{(1)}, \ldots V^{(m)}$ and witnesses $r_1, \ldots r_m$ so that when $\alpha \in (\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$ the labeling induced by the witnesses on $\mathcal{S}$ corresponds to the vector $\mathbf{x}$. This means that for $\alpha \in (\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$ the cost function $\Phi^{(p)}_{\mathcal{A}_b(\alpha)}(\mathcal{V}^{(i)})$ must be greater than r_i if the ith term in $\mathbf{x}$ is 1, and less than r_i otherwise. Since there are only $2^m = \frac{n}{8}$ $\mathbf{x}$'s, it implies that for any sample $\mathcal{V}^{(i)}$ there at most $n/8$ values of α at which we want its cost to flip above/below r_i. We can we can accomplish this using Lemma 3.3.14 by choosing $\alpha_{\mathbf{x}}$'s for which $\mathcal{V}^{(i)}$ is supposed to switch labels. In this manner, we pick each $\mathcal{V}^{(i)}$ and r_i thus creating a sample of size $\Omega(\log n)$ that is shattered by $\mathcal{H}_{\mathcal{A}_b,\texttt{clus}^{(p)}}$. □

Note: Lemma 3.3.13 assumes that the pruning step fixes a partition, and then the optimal centers can be chosen for each cluster in the partition, but points may not switch clusters even if they are closer to the center in another cluster. This is desirable, for instance, in applications which much have a balanced partition.

If it is desired that the pruning step only outputs the optimal centers, and then the clusters are determined by the Voronoi partition of the centers, we modify the proof as follows. We introduce $2n'$ more points into the clustering instance: $c_1, \ldots, c_{n'}$, and $c'_1, \ldots, c'_{n'}$. Each c_i will merge to cluster A, and each c'_i will merge to cluster A'. We set the distances so that c_i and c'_i will be the best centers for A and A' when $\alpha \in (\alpha_i, \alpha_{i+1})$. The distances are also set up so that the cost of the Voronoi tiling induced by c_{2i} and c'_{2i} is r_{low}, and the cost for c_{2i+1} and c'_{2i+1} is r_{high}. This is sufficient for the argument to go through.

Furthermore, the lower bound holds even if the cost function is the symmetric distance to the ground truth clustering. For this proof, let $A \cup \bigcup_i \{z_{2i}, z'_{2i+1}\}$ and $A' \cup \bigcup_i \{z_{2i+1}, z'_{2i}\}$ be the ground truth clustering. Then in each interval as α increases, the cost function switches between having $(n-2)/3$ errors and having $(n-2)/3 - 2$ errors.

Now we give a lower bound for $\mathcal{A}_2$.

Lemma 3.3.15. *For all objective functions $\Phi^{(p)}$, Pdim$\big(\mathcal{H}_{\mathcal{A}_2,\Phi^{(p)}}\big) = \Omega(n)$.*

Here is the intuition behind the proof. The crux of the proof is to show that there exists a clustering instance $\mathcal{V}$ over n points, a witness r, and a set of α's

$$1 = \alpha_0 < \alpha_1 < \cdots < \alpha_{2^N} < \alpha_{2^N+1} = 3,$$

where $N = \lfloor (n-8)/4 \rfloor$, such that $\Phi_{\mathcal{A}_2,\mathcal{V}}(\alpha)$ oscillates above and below r along the sequence of intervals (α_i, α_{i+1}). We finish the proof in a manner similar to Lemma 3.3.13 by constructing instances with fewer oscillations.

To construct $\mathcal{V}$, first we define two pairs of points which merge together regardless of the value of α. Call these merged pairs A and B. Next, we define a sequence of points p_i and q_i for $1 \leq i \leq N$ with distances set such that merges involving points in this sequence occur one after the other. In particular, each p_i merges with one of A or B while q_i merges with the other. Therefore, there are potentially 2^N distinct merge trees which can be created. Using induction to precisely set the distances, we show there are 2^N distinct values of α, each corresponding to a unique merge tree, thus enabling $\mathcal{A}_2$ to achieve all possible merge tree behaviors. Finally, we carefully add more points to the instance to control the oscillation of the cost function over these intervals as desired.

Now we give the full details of the proof. We start with a helper lemma.

Lemma 3.3.16. *Given n, and setting $N = \lfloor (n-8)/2 \rfloor$, then there exists a clustering instance $\mathcal{V} = (V, d)$ of size $|V| = n$ and a set of $2^N + 2$ values of α for which α-linkage creates a unique merge tree.*

Proof. The idea of the proof is as follows. First we define two pairs of points which merge together regardless of the value of α. Call these merged pairs $A = \{p_a, q_a\}$ and $B = \{p_b, q_b\}$. Next, we define a sequence of points p_i and q_i for $1 \leq i \leq N$ with distances set such that merges involving points in this sequence occur one after the other. In particular, first p_1 merges to A or B, then q_1 merges to the opposite set, then p_2 merges to A or B and q_2 merges to the opposite set, and so on. Using induction to precisely set all the distances, we show that for all $1 \leq i \leq N$, p_i merges to A or B based on the value of α, regardless of all previous merges that took place. Therefore, there are 2^N distinct merge trees which can be created. In particular, there are 2^N distinct values of α, each corresponding to a distinct merge tree, enabling $\mathcal{A}_2$ to achieve all possible merge tree behaviors. Finally, we carefully add more points to the instance to control the oscillation of the cost function over these intervals as desired.

Now we go into more detail on the specifics of the construction. We set the distances so the first two merges will always be p_a to q_a, and p_b to q_b. These sets $\{p_a, q_a\}$ and $\{p_b, q_b\}$ will stay separated until the last few merge operations. Throughout the analysis, at any point in the merging procedure, we denote the current superset containing $\{p_a, q_a\}$ by A, and we similarly denote the superset of $\{p_b, q_b\}$ by B. Next, we construct the distances so that p_i and q_i will always merge before p_j and q_j, for $i < j$. Furthermore, for all i, $\{p_i\}$ will first merge to A or B, and then $\{q_i\}$ will merge to the other one. We call these merges 'round i', for $1 \leq i \leq N$. Finally, there will be a set C_A of size $N + 2$ which merges together and then merges to A, and similarly a set C_B which

merges to B. These sets will control the value of the resulting clusterings. In our construction, the only freedom is whether p_i merges to A or to B, for all i, which is 2^N combinations total. The crux of the proof is to show there exists a unique α for each of these behaviors.

In round 1, the following equation specifies whether p_1 merges to A or B:

$$\frac{1}{2}(d(p_a, p_1)^\alpha + d(q_a, p_1)^\alpha) \lesseqgtr \frac{1}{2}(d(p_b, p_1)^\alpha + d(q_b, p_1)^\alpha)$$

If the LHS is smaller, then p_1 merges to A, otherwise it merges to B. We set the distances to ensure there exists a value α' which is the only solution to the equation in the range $(1, 3)$. Then p_1 merges to A for all $\alpha \in (1, \alpha')$, and B for all $\alpha \in (\alpha', 3)$. We set $d(p_1, q_1)$ to be large, so that once p_1 merges to either A or B, q_1 is forced to the other set, the one which does not contain p_1.

In round 2, there are two equations:

$$\frac{1}{3}(d(p_a, p_2)^\alpha + d(q_a, p_2)^\alpha + d(p_1, p_2)^\alpha) \lesseqgtr \frac{1}{3}(d(p_b, p_2)^\alpha + d(q_b, p_2)^\alpha + d(q_1, p_2)^\alpha),$$
$$\frac{1}{3}(d(p_a, p_2)^\alpha + d(q_a, p_2)^\alpha + d(q_1, p_2)^\alpha) \lesseqgtr \frac{1}{3}(d(p_b, p_2)^\alpha + d(q_b, p_2)^\alpha + d(p_1, p_2)^\alpha).$$

The first equation specifies where p_2 merges in the case when $p_1 \in A$, and the second equation is the case when $p_1 \in B$. So we must ensure there exists a specific $\alpha_{[-1]} \in (1, \alpha')$ which solves equation 1, and $\alpha_{[1]} \in (\alpha', 3)$ which solves equation 2, and these are the only solutions in the corresponding intervals.

In general, round i has 2^{i-1} equations corresponding to the 2^{i-1} possible states for the partially constructed tree. For each state, there is a specific α interval which will cause the algorithm to reach that state. We must ensure that the equation has exactly one solution in that interval. By achieving this simultaneously for every equation, the next round will have $2 \cdot 2^{i-1}$ states. See Figure 3.4 for a schematic of the clustering instance.

For $1 \leq i \leq N$, given $\mathbf{x} \in \{-1, 1\}^{i-1}$, let $E_{\mathbf{x}}$ denote the equation in round i which determines where p_i merges, in the case where for all $1 \leq j < i$, p_j merged to A if $x_j = -1$, or B if $x_j = 1$ (and let E' denote the single equation for round 1). Let $\alpha_{\mathbf{x}} \in (1, 3)$ denote the solution to $E_{\mathbf{x}} = 0$. Then we need to show the α's are well-defined and follow a specific ordering, shown in Figure 3.5. This ordering is completely specified by two conditions: *(1)* $\alpha_{[\mathbf{x}\ -1]} < \alpha_{[\mathbf{x}]} < \alpha_{[\mathbf{x}\ 1]}$ and *(2)* $\alpha_{[\mathbf{x}\ -1\ \mathbf{y}]} < \alpha_{[\mathbf{x}\ 1\ \mathbf{z}]}$ for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \bigcup_{i<N}\{-1, 1\}^i$ and $|\mathbf{y}| = |\mathbf{z}|$.

Now we show how to set up the distances to achieve all of these properties. In the first round, we set the distances so that the merge equation is $2 \cdot 1.1^\alpha \lesseqgtr (1.1 - q^*)^\alpha + (1.1 + q^*)^\alpha$, for some offset value q^* which solves the equation at $\alpha = 2$. Therefore, $\alpha \in (1, 2)$ corresponds to $p_1 \in A$, and $\alpha \in (2, 3)$ corresponds to $p_1 \in B$. In the next round, there are three distances on each side of the merge equations, since p_1 and q_1 are added to sets A and B. In the first case, when $p_1 \in A$, the merge equation for round 2 is $2 \cdot 1.1^\alpha + (1.5 - o_1)^\alpha \lesseqgtr (1.1 - q^*)^\alpha + (1.1 + q^*)^\alpha + (1.5 + o_1)^\alpha$, and when $p_1 \in B$ the equation is $2 \cdot 1.1^\alpha + (1.5 + o_1)^\alpha \lesseqgtr (1.1 - q^*)^\alpha + (1.1 + q^*)^\alpha + (1.5 - o_1)^\alpha$. By setting the offset small enough, we ensure that both solutions to the equations fall in their respective ranges of $(1, 2)$ and $(2, 3)$. This ensures that there are four distinct values of α, such that we get four distinct merge trees after round 2. The rest of the rounds repeat this pattern. For each

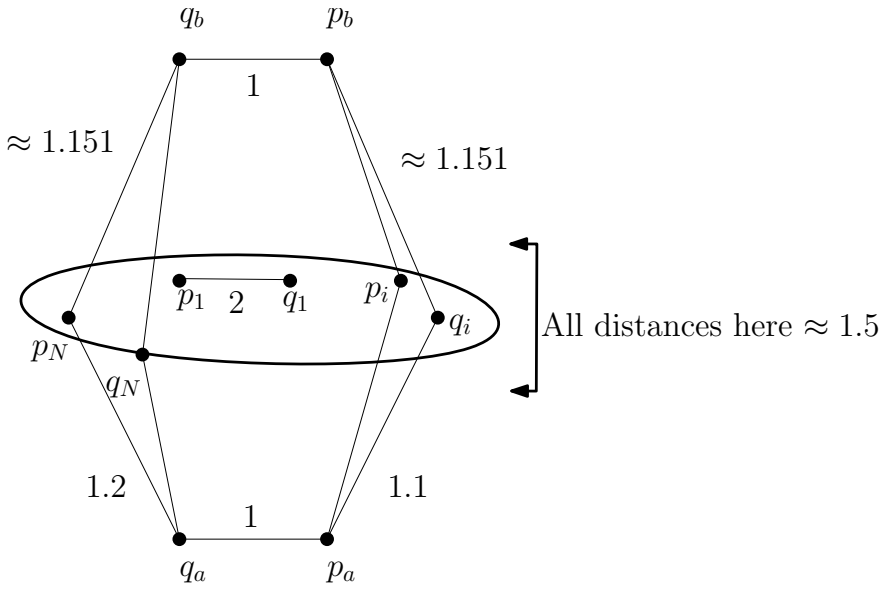


Figure 3.4: The clustering instance used in Lemma 3.3.15

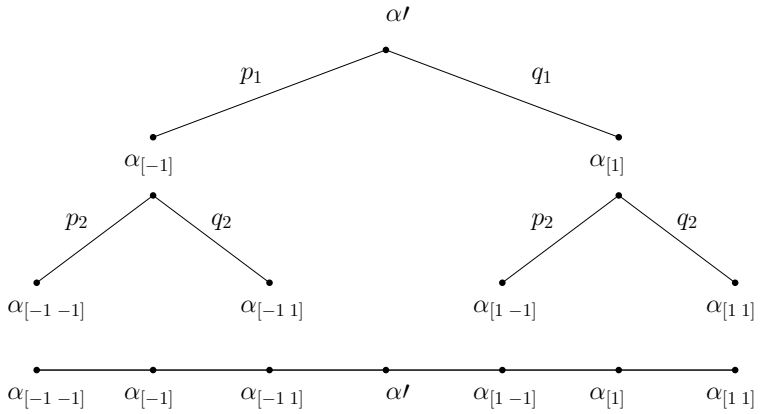


Figure 3.5: A schematic for the α intervals. Each edge denotes whether to merge p_i to A or q_i to A.

new round i, the new distances added to the equation will be $1.5 + o_i$ and $1.5 - o_i$, and we set these offsets o_i smaller and smaller so that the solutions to the equations stay in the correct ranges. To precisely show that such values of the offsets exist, we use an inductive proof.

Our inductive proof will need the following fact (true by elementary calculus).

Fact 3.3.17. *For all $0 \leq z \leq .01$ and $\alpha \in (1, 3)$, the following are true about the functions*

$$g(z, \alpha) = (1.5 - z)^\alpha - (1.5 + z)^\alpha \text{ and}$$
$$h(z, \alpha) = (1.1 - z)^\alpha + (1.1 + z)^\alpha - 2 \cdot (((1.1 - z)^\alpha + (1.1 + z)^\alpha)/2)^{\frac{\alpha}{2}}.$$

1. *For $z > 0$, $g(z, \alpha) < 0$,*
2. *for a fixed z, g is nonincreasing in α,*
3. *for a fixed α, g is nonincreasing in z,*
4. *$h(0, \alpha) = 0$ and h is nondecreasing in z.*

Here are the details for the general construction. All distances will be between 1 and 2 so that the triangle inequality is satisfied. Given N, for all i,

$$\begin{aligned}
& d(p_a, q_a) = d(p_b, q_b) = 1, \\
& d(p_a, q_a) = d(p_a, q_b) = d(p_b, q_a) = d(p_b, q_b) = 2, \\
\forall i \leq N,\ & d(p_a, p_i) = d(p_a, q_i) = 1.1 - q,\ d(q_a, p_i) = d(q_a, q_i) = 1.1 + q, \\
& d(p_b, p_i) = d(p_b, q_i) = d(q_b, p_i) = d(q_b, q_i) = \sqrt{\frac{1}{2}((1.1 - q)^2 + (1.1 + q)^2)}, \\
& d(p_i, q_i) = 2, \\
\forall 1 \leq j < i \leq N,\ & d(p_i, p_j) = d(p_i, q_j) = 1.5 + o_j \\
& d(q_i, p_j) = d(q_i, q_j) = 1.5 - o_j.
\end{aligned}$$

where q and o_j are offset values in $(0, .01)$ which we will specify later. Then for $\alpha \in (1, 3)$, the following are true.

- The first two merges are p_a to q_a and p_b to q_b,
- $\{p_i\}$ and $\{q_i\}$ will always prefer merging to A or B instead of merging to another singleton $\{p_j\}$ or $\{q_j\}$.

After the first two merges occur, all p_i and q_i are tied to first merge to A or B. For convenience, we specify the tiebreaking order as $\{p_1, q_1, \ldots, p_N, q_N\}$. Alternatively, at the end we can make tiny perturbations to the distances so that tiebreaking does not occur.

Next, we choose the value for q, which must be small enough to ensure that q_i always merges to the opposite cluster as p_i. Consider

$$h(\alpha, q, o_1, \ldots, o_N, \mathbf{x}) = \frac{N+2}{N+3}\left((1.1+q)^\alpha + (1.1-q)^\alpha + \sum_{i<N} \mathbf{x}_i(1.5+o_i)^\alpha + 1.5^\alpha\right) - 2\cdot(((1.1+q)^2 + (1.1-q)^2)/2)^{\frac{\alpha}{2}} - \sum_{i<N} \mathbf{x}_i(1.5+o_i)^\alpha.$$

If this equation is positive for all $\mathbf{x} \in \{-1,1\}^{N-1}$, then q_N will always merge to the opposite cluster as p_N (and q_i will always merge to the opposite cluster as p_i, which we can similarly show by setting $o_j = 0$ in h for all $j > i$).

Note

$$h(\alpha, 0, 0, \ldots, 0, \mathbf{x}) = \frac{N+2}{N+3}\left(2\cdot 1.1^\alpha + (N+1)\cdot 1.5^\alpha\right) - 2\cdot 1.1^\alpha - N\cdot 1.5^\alpha > 0$$

for all $\mathbf{x}$ and all $\alpha \in (1,3)$. Fact 3.3.17 implies there exists a $0 < q^* < .01$ such that $h(\alpha, q, 0, \ldots, 0, \mathbf{x})$ stays positive. Similarly, there exists a cutoff value $\delta > 0$ such that for all $0 < o_1, \ldots, o_N < \delta$, $\alpha \in (1,3)$, and $\mathbf{x} \in \{-1,1\}^{N-1}$, $h(\alpha, q^*, o_1, \ldots, o_k, \mathbf{x}) > 0$. Therefore, as long as we set all the offsets o_i less than δ, the merges will be as follows:

1. p_a merges to q_a and p_b merges to q_b.
2. For $1 \ldots, N$, p_i merges to A or B, and q_i merges to the opposite cluster. Then q_N will always merge to the opposite cluster as p_N.

Now we show that there are 2^N intervals for $\alpha \in (1,3)$ which give unique behavior. Recall for $\mathbf{x} \in \bigcup_{i<N}\{-1,1\}^i$, $E_{\mathbf{x}}$ is defined as

$$(1.1-q^*)^\alpha + (1.1+q^*)^\alpha - 2\cdot(\frac{1}{2}((1.1-q^*)^2 + (1.1+q^*)^2))^{\frac{\alpha}{2}} + \sum_{i<N} \mathbf{x}_i((1.5-o_i)^\alpha - (1.5+o_i)^\alpha).$$

For brevity, we denote

$$d = (\frac{1}{2}((1.1-q^*)^2 + (1.1+q^*)^2))^{\frac{1}{2}}.$$

We show the αs are correctly ordered by proving the following three statements with induction. The first statement is sufficient to order the αs, and the second two will help to prove the first.

1. There exist $0 < o_1, \ldots, o_N < \delta$ such that if we solve $E_{\mathbf{x}} = 0$ for $\alpha_{\mathbf{x}}$ for all $\mathbf{x} \in \bigcup_{i<N}\{-1,1\}^i$, then the α's satisfy $\alpha_{[\mathbf{x}\ -1]} < \alpha_{[\mathbf{x}]} < \alpha_{[\mathbf{x}\ 1]}$ and for all $i < N$, $\alpha_{[\mathbf{x}\ 1]} < \alpha_{[\mathbf{y}\ -1]}$ for $\mathbf{x}, \mathbf{y} \in \{-1,1\}^i$ and $\mathbf{x}_1 \ldots \mathbf{x}_i < \mathbf{y}_1 \ldots \mathbf{y}_i$.
2. For all $k' \leq N$ and $\alpha \in (1,3)$,

$$(1.5+o_{k'})^\alpha - (1.5-o_{k'})^\alpha + \sum_{k'<i<N} ((1.5-o_i)^\alpha - (1.5+o_i)^\alpha) > 0.$$

3.

$$(1.1-q^*)^3+(1.1+q^*)^3-2\cdot d^3+\sum_{i<N}((1.5-o_i)^3-(1.5+o_i)^3)>0, \text{ and}$$
$$(1.1-q^*)+(1.1+q^*)-2\cdot d+\sum_{i<N}((1.5+o_i)-(1.5-o_i))<0.$$

We proved the base case in our earlier example for $n=10$. Assume for $k\leq N$, there exist $0<o_1,\ldots,o_k<\delta$ which satisfy the three properties. We first prove the inductive step for the second and third statements.

By inductive hypothesis, we know for all $k'\leq k$ and $\alpha\in(1,3)$,

$$(1.5+o_{k'})^\alpha-(1.5-o_{k'})^\alpha+\sum_{k'<i\leq k}((1.5-o_i)^\alpha-(1.5+o_i)^\alpha)>0,$$

Since there are finite integral values of $k'\leq k$, and the expression is >0 for all values of k', then there exists an $\epsilon>0$ such that the expression is $\geq\epsilon$ for all values of k'. Then we define z_a such that $(1.5+z_a)^\alpha-(1.5-z_a)^\alpha<\frac{\epsilon}{2}$ for $\alpha\in(1,3)$. Then for all $0<z<z_a$, $k'\leq k+1$, and $\alpha\in(1,3)$,

$$(1.5+o_{k'})^\alpha-(1.5-o_{k'})^\alpha+\sum_{k'<i\leq k+1}((1.5-o_i)^\alpha-(1.5+o_i)^\alpha)>0.$$

So as long as we set $0<o_{k+1}<z_a$, the inductive step of the second property will be fulfilled. Now we move to the third property. We have the following from the inductive hypothesis:

$$(1.1-q^*)^3+(1.1+q^*)^3-2\cdot d^3+\sum_{i\leq k'}((1.5-o_i)^3-(1.5+o_i)^3)>0,$$
$$(1.1-q^*)+(1.1+q^*)-2\cdot d+\sum_{i\leq k'}((1.5+o_i)-(1.5-o_i))<0.$$

We may similarly find z_b such that for all $0<o_{k+1}<z_b$,

$$(1.1-q^*)^3+(1.1+q^*)^3-2\cdot d^3+\sum_{i\leq k+1}((1.5-o_i)^3-(1.5+o_i)^3)>0,$$
$$(1.1-q^*)+(1.1+q^*)-2\cdot d+\sum_{i\leq k+1}((1.5+o_i)-(1.5-o_i))<0.$$

Now we move to proving the inductive step of the first property. Given $\mathbf{x}\in\{-1,1\}^k$, let $p(\mathbf{x}),n(\mathbf{x})\in\{-1,1\}^k$ denote the vectors which sit on either side of $\alpha_{\mathbf{x}}$ in the ordering, i.e., $\alpha_{\mathbf{x}}$ is the only $\alpha_{\mathbf{y}}$ in the range $(\alpha_{p(\mathbf{x})},\alpha_{n(\mathbf{x})})$ such that $|\mathbf{y}|=k$. If $\mathbf{x}=[1\ \ldots\ 1]$, then set $\alpha_{n(\mathbf{x})}=3$, and if $\mathbf{x}=[0\ \ldots\ 0]$, set $\alpha_{p(\mathbf{x})}=1$. Define

$$f(\alpha,\mathbf{x},z)=E_{\mathbf{x}}+(1.5-z)^\alpha-(1.5+z)^\alpha.$$

By inductive hypothesis, we have that $f(\alpha_{\mathbf{x}}, \mathbf{x}, 0) = 0$. We must show there exists $z_{\mathbf{x}}$ such that for all $0 \leq z \leq z_{\mathbf{x}}$, $f(\alpha_{\mathbf{x}}, \mathbf{x}, z) < 0$ and $f(\alpha_{n(\mathbf{x})}, \mathbf{x}, z) > 0$. This will imply that if we choose $0 < o_{k+1} < z_{\mathbf{x}}$, then $\alpha_{[\mathbf{x}\ 1]} \in (\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$.

Case 1: $\mathbf{x} \neq [1 \dots 1]$. Since $f(\alpha_{\mathbf{x}}, \mathbf{x}, 0) = 0$, and by Fact 3.3.17, then for all $0 < z < .01$, $f(\alpha_{\mathbf{x}}, \mathbf{x}, z) < 0$. Now denote i^* as the greatest index such that $\mathbf{x}_{i^*} = -1$. Then $n(\mathbf{x}) = [\mathbf{x}_1 \dots \mathbf{x}_{i^*-1}\, 1\, -1 \cdots -1]$. By statement 1 of the inductive hypothesis ($\alpha_{n(\mathbf{x})}$ is a root of $E_{n(\mathbf{x})} = 0$),

$$(1.1 - q^*)^{\alpha_{n(\mathbf{x})}} + (1.1 + q^*)^{\alpha_{n(\mathbf{x})}} - 2 \cdot d^{\alpha_{n(\mathbf{x})}} + \sum_{i \leq k} (n(\mathbf{x})_i (1.5 - o_i)^{\alpha_{n(\mathbf{x})}} - n(\mathbf{x})_i (1.5 + o_i)^{\alpha_{n(\mathbf{x})}}) = 0.$$

From statement 2 of the inductive hypothesis, we know that

$$(1.5 - o_{i^*})^{\alpha_{n(\mathbf{x})}} - (1.5 + o_{i^*})^{\alpha_{n(\mathbf{x})}} + \sum_{i^* < i \leq k} ((1.5 + o_i)^{\alpha_{n(\mathbf{x})}} - (1.5 - o_i)^{\alpha_{n(\mathbf{x})}}) < 0.$$

It follows that

$$(1.1 - q^*)^{\alpha_{n(\mathbf{x})}} + (1.1 + q^*)^{\alpha_{n(\mathbf{x})}} - 2 \cdot d^{\alpha_{n(\mathbf{x})}} + \sum_{i < i^*} (n(\mathbf{x})_i (1.5 - o_i)^{\alpha_{n(\mathbf{x})}} - n(\mathbf{x})_i (1.5 + o_i)^{\alpha_{n(\mathbf{x})}}) > 0,$$

and furthermore,

$$(1.1 - q^*)^{\alpha_{n(\mathbf{x})}} + (1.1 + q^*)^{\alpha_{n(\mathbf{x})}} - 2 \cdot d^{\alpha_{n(\mathbf{x})}} + \sum_{i < i^*} (\mathbf{x}_i (1.5 - o_i)^{\alpha_{n(\mathbf{x})}} - \mathbf{x}_i (1.5 + o_i)^{\alpha_{n(\mathbf{x})}}) > 0.$$

Therefore, $f(\alpha_{n(\mathbf{x})}, 0) > 0$, so denote $f(\alpha_{n(\mathbf{x})}, 0) = \epsilon > 0$. Then because of Fact 3.3.17, there exists $z_{\mathbf{x}}$ such that $\forall 0 < z < z_{\mathbf{x}}$, $f(\alpha_{n(\mathbf{x})}, z) > 0$.

Case 2: $\mathbf{x} = [1 \dots 1]$. Since $f(\alpha_{\mathbf{x}}, 0) = 0$, and by Fact 3.3.17, then for all $0 < z < .01$, $f(\alpha_{\mathbf{x}}, z) < 0$. By property 3 of the inductive hypothesis, we have

$$(1.1 - q^*)^3 + (1.1 + q^*)^3 - 2 \cdot d^3 + \sum_{i \leq k} ((1.5 - o_i)^3 - (1.5 + o_i)^3) > 0,$$

so say this expression is equal to some $\epsilon > 0$. Then from Fact 3.3.17, there exists $z_{\mathbf{x}}$ such that for all $0 < z < z_{\mathbf{x}}$, $0 < (1.5 + z)^3 - (1.5 - z)^3 < \frac{\epsilon}{2}$. Combining these, we have $f(3, z) > 0$ for all $0 < z < z_{\mathbf{x}}$.

To recap, in both cases we showed there exists $z_{\mathbf{x}}$ such that for all $0 < z < \min(.01, z_{\mathbf{x}})$, $f(\alpha_{\mathbf{x}}, z) < 0$ and $f(\alpha_{n(\mathbf{x})}, z) > 0$. We may perform a similar analysis on a related function f', defined as $f'(\alpha, \mathbf{x}, z) = E_{\mathbf{x}} + (1.5 + z)^{\alpha} - (1.5 - z)^{\alpha}$ to show there exists $z'_{\mathbf{x}}$ such that for all $0 < z < z'_{\mathbf{x}}$, $f'(\alpha_{p(\mathbf{x})}, z) < 0$ and $f'(\alpha_{\mathbf{x}}, z) > 0$. We perform this analysis over all $\mathbf{x} \in \{-1, 1\}^k$.

Finally, we set $o_{k+1} = \min_{\mathbf{x}}(z_{\mathbf{x}}, z'_{\mathbf{x}}, z_a, z_b, .01)$. Given $\mathbf{x} \in \{-1, 1\}^k$, since $f(\alpha_{\mathbf{x}}, o_{k+1}) < 0$ and $f(\alpha_{n(\mathbf{x})}, o_{k+1}) > 0$, there must exist a root $\alpha_{[\mathbf{x}\ 1]} \in (\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$ (and by Fact 3.3.17, the function is monotone in α in the short interval $(\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$, so there is exactly one root). Similarly, there must exist a root $\alpha_{[\mathbf{x}\ -1]} \in (\alpha_{p(\mathbf{x})}, \alpha_{\mathbf{x}})$. Then we have shown $\alpha_{[\mathbf{x}\ -1]}$ and $\alpha_{[\mathbf{x}\ 1]}$ are roots of $E_{[\mathbf{x}\ -1]}$ and $E_{[\mathbf{x}\ 1]}$, respectively. By construction, $\alpha_{[\mathbf{x}\ -1]} < \alpha_{\mathbf{x}} < \alpha_{[\mathbf{x}\ 1]}$, so condition 1 is satisfied. Now we need to show condition 2 is satisfied. Given $\mathbf{x}, \mathbf{y} \in \{-1, 1\}^k$, let k' be the largest number for which $\mathbf{x}_i = \mathbf{y}_i$, $\forall i \leq k'$. Let $\mathbf{z} = \mathbf{x}_{[1 \dots k']} = \mathbf{y}_{[1 \dots k']}$. Then by the inductive hypothesis,

$$\alpha_{\mathbf{x}} < \alpha_{n(\mathbf{x})} \leq \alpha_{\mathbf{z}} \leq \alpha_{p(\mathbf{y})} < \alpha_{\mathbf{y}}.$$

It follows that

$$\alpha_{[\mathbf{x}\ -1]} < \alpha_{[\mathbf{x}\ 1]} < \alpha_{\mathbf{z}} < \alpha_{[\mathbf{y}\ -1]} < \alpha_{[\mathbf{y}\ 1]},$$

proving condition 2. This completes the induction. □

Now we are ready to prove Lemma 3.3.15.

Proof of Lemma 3.3.15. Given n, and setting $N = \lfloor (n-8)/4 \rfloor$, we will show there exists a clustering instance (V, d) of size $|V| = n$, a witness r, and a set of $2^N + 2$ α's $1 = \alpha_0 < \alpha_1 < \cdots < \alpha_{2^N} < \alpha_{2^N+1} = 3$, such that $\Phi^{(p)}_{\mathcal{A}_3(\alpha)}(\mathcal{V})$ oscillates above and below r between each interval (α_i, α_{i+1}).

We start by using the construction from Lemma 3.3.15, which gives a clustering instance with $2N + 8$ points and $2^N + 2$ values of α for which α-linkage creates a unique merge tree. The next part is to add $2N$ more points and define a witness r so that the cost function alternates above and below r along each neighboring α interval, for a total of 2^N oscillations. Finally, we will finish off the proof in a manner similar to Lemma 3.3.13.

Starting with the clustering instance (V, d) from Lemma 3.3.15, we add two sets of points, C_A and C_B, which do not interfere with the previous merges, and ensure the cost functions alternates. Let $C_A = \{c_a, c'_a, a_1, a_2, \ldots, a_N\}$ and $C_B = \{c_b, c'_b, b_1, b_2, \ldots, b_N\}$. All distances between two points in C_A are 1, and similarly for C_B. All distances between a point in C_A and a point in C_B are 2. The distances between $C_A \cup C_B$ and $A \cup B$ are as follows (we defined the sets A and B in Lemma 3.3.15).

$$\begin{aligned}
&d(p_a, c_a) = d(p_a, c'_a) = d(q_a, c_a) = d(q_a, c'_a) = 1.51,\\
&d(p_b, c_b) = d(p_b, c'_b) = d(q_b, c_b) = d(q_b, c'_b) = 1.51,\\
&d(p_a, c_b) = d(p_a, c'_b) = d(q_a, c_b) = d(q_a, c'_b) = 2,\\
&d(p_b, c_a) = d(p_b, c'_a) = d(q_b, c_a) = d(q_b, c'_a) = 2,\\
&d(p_a, c) = d(q_a, c) = d(p_b, c) = d(q_b, c) = 2\ \forall c \in C_A \cup C_B \setminus \{c_a, c'_a, c_b, c'_b\},\\
&d(c, p_i) = d(c, q_i) = 1.51\ \forall 1 \le i \le N-1 \text{ and } c \in C_A \cup C_B.
\end{aligned}$$

We will specify the distances between $\{c_a, c'_a, c_b, c'_b\}$ and $\{p_N, q_N\}$ soon, but they will be in $[1.6, 2]$. So at the start of the merge procedure, all points in C_A merge together, and all points in C_B merge together. Then all merges from Lemma 3.3.15 take place, because all relevant distances are smaller than 1.51. We end up with four sets: A, B, C_A, and C_B. The pairs (A, B) and (C_A, C_B) are dominated by distances of length 2, so the merges (C_A, A) and (C_B, B) will occur, which dominate (C_A, B) and (C_B, A) because of the distances between $\{p_a, q_a, p_b, q_b\}$ and $\{c_a, c'_a, c_b, c'_b\}$. The final merge to occur will be $(C_A \cup A, C_B \cup B)$, however, the 2-median pruning step will clearly pick the 2-clustering $C_A \cup A$, $C_B \cup B$, since no other clustering in the tree has almost all distances ≤ 1.51. Then by construction, c_a or c'_a will be the best center for $C_A \cup A$, which beat p_a and q_a because $1.51 \cdot (2N) < 1.1 \cdot N + 2 \cdot N = 1.55 \cdot (2N)$. Similarly, c_b or c'_b will be the best center for $C_B \cup B$. Note that centers $\{c_a, c'_a\}$ and $\{c_b, c'_b\}$ currently give equivalent 2-median costs. Denote this cost by r' (i.e., the cost before we set the distances to p_N and q_N).

Now we set the final distances as follows.

$$\begin{aligned} d(c_a, p_N) &= d(c_b, q_N) = 1.6, \\ d(c'_a, p_N) &= d(c'_b, q_N) = 1.7, \\ d(c'_a, q_N) &= d(c'_b, p_N) = 1.8, \\ d(c_a, q_N) &= d(c_b, p_N) = 1.9. \end{aligned}$$

If $p_N \in A$ and $q_N \in B$, then c_a and c_b will be the best centers, achieving cost $r' + 3.2$ for $(C_A \cup A, C_B \cup B)$. If $p_N \in B$ and $q_N \in A$, then c'_a and c'_b will be the best centers, achieving cost $r' + 3.6$ for $(C_A \cup A, C_B \cup B)$.

The distances are also constructed so that in the variant where the pruning outputs the optimal centers, and then all points are allowed to move to their closest center, the cost still oscillates. First note that no points other than p_N and q_N are affected, since $d(c_a, p_i) = d(c_a, q_i)$ for $i < N$, and similarly for c_b. Then p_N will move to the cluster with c_a or c'_a, and q_N will move to the cluster with c_b or c'_b. If p_N was originally in A, then the cost is $r' + 3.2$, otherwise the cost is $r' + 3.4$.

In either scenario, we set $r = r' + 3.3$. Then we have ensured for all $\mathbf{x} \in \{-1, 1\}^{N-1}$, the cost for $\alpha \in (\alpha_{p(\mathbf{x})}, \alpha_{\mathbf{x}})$ is $< r$, and the cost for $\alpha \in (\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$ is $> r$. We have finished our construction of a clustering instance whose cost function alternates 2^N times as α increases.

To finish the proof, we will show there exists a set $S = \{V_1, \ldots, V_s\}$ of size $s = N = \lfloor (n-8)/4 \rfloor \in \Omega(n)$ that is shattered by $\mathcal{A}$. Such a set has 2^N orderings total. For V_1, we use the construction which alternates 2^N times. For V_2, we use the same construction, but we eliminate (p_N, q_N) so that there are only $N - 1$ rounds (the extra two points can be added to C_A and C_B to preserve $|V_2| = n$). Then V_2's cost will alternate $\frac{1}{2} \cdot 2^N$ times, between the intervals $(\alpha_{p(\mathbf{x})}, \alpha_{\mathbf{x}})$ and $(\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$, for $\mathbf{x} \in \{-1, 1\}^{N-2}$. So V_2 oscillates every other time V_1 oscillates, as α increases. In general, V_i will be the construction with only $N - i + 1$ rounds, oscillating $2^{\frac{N}{2^{i-1}}}$ times, and each oscillation occurs every other time V_{i-1} oscillates. This ensures for every $\mathbf{x} \in \{-1, 1\}^{N-1}$, $(\alpha_{p(\mathbf{x})}, \alpha_{\mathbf{x}})$ and $(\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$ will have unique labelings, for a total of 2^N labelings. This completes the proof. □

Note: As in Lemma 3.3.13, this lower bound holds even if the cost function is the symmetric distance to the ground truth clustering. Merely let p_N and q_N belong to different ground truth clusters, but for all $i < N$, p_i and q_i belong to the same ground truth cluster. Since in each adjacent α interval, p_N and q_N switch clusters, this shows the symmetric distance to the ground truth clustering oscillates between every interval.

Furthermore, as was the case in Lemma 3.3.14, we can achieve a tradeoff between the number of oscillations, and the difference in cost between the oscillations. Specifically, for all $0 < \gamma <\leq 1$, we can show an instance which oscillates 2^N times above r and below γr, where $N = \lfloor \gamma(n-8)/4 \rfloor$. We use N points to create the gadgets above, and then we add $\frac{1-\gamma}{2} \cot n$ points to a, and $\frac{1-\gamma}{2} \cdot n$ points to z_N.

Even though the pseudo-dimension for $\mathcal{A}_1$ is tight, the runtime of Algorithm 7 might be as large as $\Theta(n^8)$, since there are $\Theta(n^8)$ decision points in the worst case. Note that the construction in

Lemma 3.3.13 has $\Omega(n)$ decision points. Next, we give a stronger lower bound of $\Omega(n^5)$ decision points, as well as intuition for extending it to $\Omega(n^7)$.

We are interested in computing the minimum number of intervals it takes to divide up $[0,1]$ such that the output of α-linkage is fixed for all α over a distinct interval. Given a clustering instance $\mathcal{V} = (S, d)$ of size $|S| = n$, we denote the number of intervals as $\#I$. We can also think of $\#I$ as the number of discontinuities of the cost function $\text{cost}(\mathcal{V}, \alpha)$ of the clustering outputted by α-linkage, as a function of α.

Over (S, d), there are $O(n^2)$ distances. The merge value of any two sets $A, B \subseteq S$ is defined by two distances $d_{\text{min}}^{(A,B)}, d_{\text{max}}^{(A,B)}$, the minimum and maximum distances between points in A and B. Specifically, the merge value of (A, B) is

$$\alpha d_{\text{min}}^{(A,B)} + (1-\alpha) d_{\text{max}}^{(A,B)} = d_{\text{max}}^{(A,B)} - (d_{\text{max}}^{(A,B)} - d_{\text{min}}^{(A,B)})\alpha.$$

We give a few simple observations.

Fact 3.3.18. *Given two pairs of sets (A, B) and (C, D), if $d_{min}^{(A,B)} < d_{min}^{(C,D)}$ and $d_{max}^{(A,B)} < d_{max}^{(C,D)}$, then (A, B) will have a lower merge value than (C, D) for all values of α in $[0, 1]$.*

This is easy to see: $\alpha d_{\text{min}}^{(A,B)} + (1-\alpha) d_{\text{max}}^{(A,B)} < \alpha d_{\text{min}}^{(C,D)} + (1-\alpha) d_{\text{max}}^{(C,D)}$.

Fact 3.3.19. *Given two pairs of sets (A, B) and (C, D) such that $d_{min}^{(A,B)} < d_{min}^{(C,D)}$, if $d_{min}^{(A,B)} < d_{min}^{(C,D)} < d_{max}^{(C,D)} < d_{max}^{(A,B)}$, then there exists $\alpha^* \in [0, 1]$ such that (C, D) has the lower merge value on $[0, \alpha^*]$, and (A, B) has the lower merge value on $[\alpha^*, 1]$.*

Proof. Recall that (A, B) and (C, D) each define linear equations, so there exists a single intersection. When $\alpha = 0$, the merge values of (A, B) and (C, D) are $d_{\text{max}}^{(A,B)}$ and $d_{\text{max}}^{(C,D)}$, respectively, so the value for (C, D) is lower. When $\alpha = 1$, the values are $d_{\text{min}}^{(A,B)}$ and $d_{\text{min}}^{(C,D)}$, so (A, B) is lower. Therefore, the intersection of the two linear equations must be in $[0, 1]$. In fact, the value of α^* is the following.

$$\alpha^* = \frac{d_{\text{max}}^{(C,D)} - d_{\text{max}}^{(A,B)}}{d_{\text{max}}^{(C,D)} - d_{\text{max}}^{(A,B)} + d_{\text{min}}^{(A,B)} - d_{\text{min}}^{(C,D)}}. \tag{3.1}$$

□

Now we move to the $\Omega(n^5)$ lower bound. First we give an overview behind the construction. Recall the *execution tree* of α-linkage run on clustering instance $\mathcal{V}$. This is a tree where each node is labeled by a state (i.e., a partial tree of merges that have happened so far) and the set $A \subset [0, 1]$ of α values that would result in the algorithm choosing this sequence of merges. For example, assume the algorithm is in state S defined by $[\alpha_1, \alpha_2]$ and has created the partial merge tree T so far. If in the next round, the algorithm chooses merge (A, B) for all $\alpha \in [\alpha_1, \alpha_2]$, then there are no new *breakpoints* created. However, if the algorithm chooses merge (A, B) for $\alpha \in [\alpha_1, \alpha']$ and merge (C, D) for $\alpha \in [\alpha', \alpha_2]$, then the state S splits into two nodes.

In our construction every state will have at most two children. In other words, at any state in the algorithm, the are only two candidate merges the can take place. Recall that the decision

between two merges (A, B) or (C, D) depends on eight points. In the n^5 construction, we will only consider merges with the following form: a point w merges to set $A = \{x, y\}$ or $B = \{z, v\}$. The distances associated with this merge are $d(w, x)$, $d(w, y)$, $d(w, z)$, and $d(w, v)$. Since the merge equation is defined by five points, there are n^5 possible such equations. The construction will use five phases. The first four phases are 'setup', where the goal is to construct a (partial) execution tree of depth $\frac{2n}{5}$ with $\Omega(n^4)$ states. Each state will contain a unique pair of sets A and B. Then phase 5 consists of $\frac{n}{10}$ rounds where points $w_1, \ldots, w_{\frac{n}{10}}$ merge to either A or B. A new breakpoint is created in each round and each state, for a total of $\Omega(n^4) \cdot \frac{n}{10} = \Omega(n^5)$ breakpoints.

Lemma 3.3.20. *There exists a constant c such that for all $n \geq 1$, there exists a clustering instance $\mathcal{V}$ with $\#I \geq c \cdot n^5$.*

Proof. The n merges are split up into five phases of $n' = n/10$ merges each. Each phase adds a factor of n' to the number of discontinuities of $\text{cost}(\mathcal{V}, \alpha)$. In the first phase, the discontinuities added are at $\frac{1}{n'}, \frac{2}{n'}, \ldots, \frac{n'-1}{n'}$, and then in each successive phase, each interval is split up into n' new equally sized intervals. The discontinuities at the end of phase 5 are $\frac{1}{n'^5}, \frac{2}{n'^5}, \ldots, \frac{n'^5-1}{n'^5} = \frac{10^5}{n^5}, \frac{2 \cdot 10^5}{n^5}, \ldots, \frac{n-10^5}{n^5}$.

There are three main sets of points, A, B, and C. Each phase has n' points associated with it. Points $x_1, \ldots, x_{n'}$, $y_1, \ldots, y_{n'}$, $z_1, \ldots, z_{n'}$, $v_1, \ldots, v_{n'}$, and $w_1, \ldots, w_{n'}$ are associated with phases 1,2,3,4, and 5, respectively. At the start, $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$, and $C = \{c_1, c_2\}$, and each of the five phases consist of the associated points merging to either A, B, or C. In phase 1, points $x_1, \ldots, x_{n'}$ each successively merge to either A or B. In phase 2, $y_1, \ldots, y_{n'}$ merge to A or B. In phase 3, $z_1, \ldots, z_{n'}$ merge to A or C. Phases 4 and 5 have $v_1, \ldots, v_{n'}$ and $w_1, \ldots, w_{n'}$ merging to A, B, or C, respectively. Now we walk through the details of each phase.

The breakpoints of phase 1 are $\alpha_1 = \frac{1}{n'}, \ldots, \alpha_{n'-1} = \frac{n'-1}{n'}$. Within phase 1, first x_1 merges to A or B, then x_2 merges to A or B, and so on. Specifically, x_i merges to A if $\alpha < \alpha_i$, otherwise x_i merges to B. See Figure 3.6.

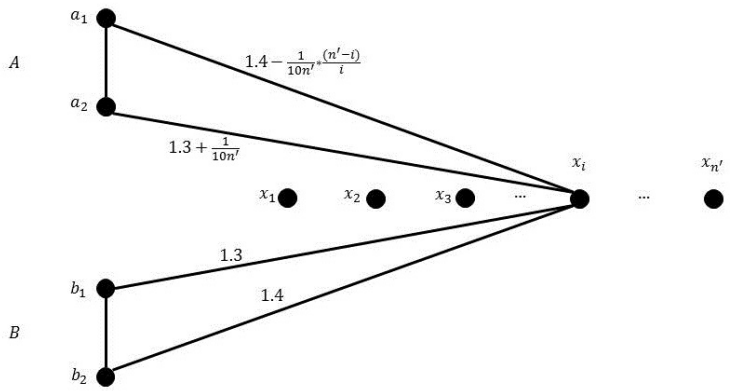


Figure 3.6: Setup of Phase 1 of Lemma 3.3.20

Therefore, we can characterize the *execution tree* at the end of phase 1 as follows: for all $1 \leq i \leq n'$, there is a node with α-interval $[\frac{i}{n'}, \frac{i+1}{n'}]$, such that $x_1, \ldots, x_i$ merged to A, and $x_{i+1}, \ldots, x_{n'}$ merged to B. See Figure 3.7.

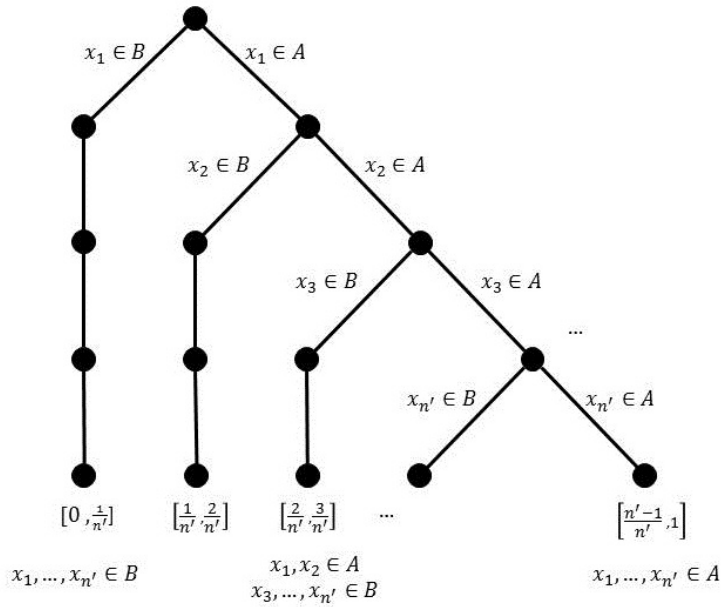


Figure 3.7: Execution tree of Phase 1 of Lemma 3.3.20

To achieve these merges, we simply set $d(b_1, x_i) = 1.3$, $d(b_2, x_i) = 1.4$, $d(a_1, x_i) = 1.3 + \frac{1}{10n'}$, and $d(a_2, x_i) = 1.4 - \frac{1}{10n'} \cdot \frac{n-i}{i} = 1.4 - \frac{1}{10} \cdot \frac{1-\alpha_i}{\alpha_i}$ for all $1 \leq i \leq n'$. Then, x_i will merge to A if $\alpha < \alpha_i$, otherwise x_i will merge to B.

Now we move to phase 2. The breakpoints of phase 2 are $\alpha_{ij} = \frac{i}{n'} + \frac{j}{n'^2}$, for all $1 \leq i, j \leq n'$. Note that for each state $[\frac{i}{n'}, \frac{i+1}{n'}]$, there will be n' new breakpoints inside this interval by the end of phase 2, for a total of n'^2 intervals. Intuitively, for each state $[\frac{i}{n'}, \frac{i+1}{n'}]$, which contains $A = \{a_1, a_2, x_1, \ldots, x_i\}$ and $B = \{b_1, b_2, x_{i+1}, \ldots, x_{n'}\}$, we will use a construction similar to phase 1 to obtain n' new breakpoints (see Figure 3.8).

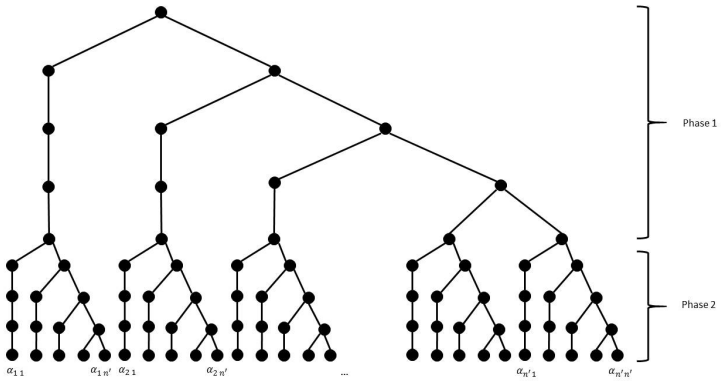


Figure 3.8: Execution tree of Phase 2

Within $[\frac{i}{n'}, \frac{i+1}{n'}]$, y_j merges to A if $\alpha < \alpha_{ij}$, otherwise y_j merges to B. The execution tree at the end of phase 2 is as follows: for all $1 \leq i, j \leq n'$, there is a node with α-interval $[\frac{i}{n'} + \frac{j}{n'^2}, \frac{i}{n'} + \frac{j+1}{n'^2}]$,

such that $x_1, \ldots, x_i, y_1, \ldots, y_j$ merged to A, and $x_{i+1}, \ldots, x_{n'}, y_{j+1}, \ldots, y_{n'}$ merged to B.

Intuitively, we can achieve these n'^2 merges by varying the distances between each pair x_i and y_j so that the breakpoints fall into the correct intervals. Specifically, for each $1 \leq i, j \leq n'$, we set $d(b_1, y_j) = 1.3$, $d(b_2, y_j) = 1.4$, $d(a_1, y_j) = 1.3 + \frac{1}{10n'}$, and $d(x_i, y_j) = 1.4 - \frac{1}{10n'} \cdot \frac{1-\alpha_{ij}}{\alpha_{ij}}$. This ensures that in state $[\frac{i}{n'}, \frac{i+1}{n'}]$, y_j will merge to A if $\alpha < \alpha_{ij}$, otherwise it will merge to B.

Before we continue with the details for phases 3, 4, and 5, we give more high level intuition. Recall that the value of α at the breakpoint is computed by the four distances in Equation 3.1. In phase 1, the decision of whether to merge x_i to A or B depends on the minimum and maximum distances from x_i to A and B, which were $d(a_1, x_i)$, $d(a_2, x_i)$, $d(b_1, x_i)$, and $d(b_2, x_i)$. In phase 2, if we had followed the same pattern, the distances would be $d(a_1, y_j)$, $d(a_2, y_j)$, $d(b_1, y_j)$, and $d(b_2, y_j)$. Instead, we replaced one of the distances with $d(x_i, y_j)$, so that the resulting breakpoint would depend on both i and j. Since there are four distances, we can continue this pattern for three more rounds.

In phase 3, the four distances will be $d(x_i, z_k)$, $d(y_j, z_k)$, $d(b_1, z_k)$, and $d(b_2, z_k)$, so the breakpoint depends on i, j, and k. In phase 4, the distances will be $d(x_i, v_\ell)$, $d(y_j, v_\ell)$, $d(z_k, v_\ell)$, and $d(b_2, v_\ell)$. Finally, in phase 5, all four distances will depend on a variable from the previous round: $d(x_i, w_m)$, $d(y_j, w_m)$, $d(z_k, w_m)$, and $d(v_\ell, w_m)$. By carefully setting all the distances, we can ensure that each equation has a solution in the correct range to add a new breakpoint, so the total number of breakpoints will be n'^5, one for each 5-tuple of (i, j, k, ℓ, m). The breakpoints will be $\alpha_{ijk\ell m} = \frac{i}{n'} + \frac{j}{n'^2} + \frac{k}{n'^3} + \frac{\ell}{n'^4} + \frac{m}{n'^5}$.

Thus, the only remaining step of the proof is to define all distances to show this construction is achievable. There are a few technicalities that must be dealt with. For example, instead of deciding whether to merge to A or B in all 5 phases, we need to introduce set C, and in some of the phases, the decision is between sets A and C or B and C. We need to introduce the set C so that the distances in the equations are properly 'nested', e.g., we need to satisfy the inequalities in Fact 3.3.19 even to ensure the value of α^* falls in $[0, 1]$. In the rest of the proof, we lay out the details for phases 3, 4, and 5.

In phase 3, we set aside set B. The decisions in phase 3 are whether to merge $z_1, \ldots, z_{n'}$ to A or $C = \{c_1, c_2\}$. At the end of phase 2, there are n'^2 states: for all $1 \leq i, j \leq n'$, $A = \{a_1, a_2, x_1, \ldots, x_i, y_1, \ldots, y_j\}$ for $\alpha \in [\alpha_{ij}, \alpha_{ij} + \frac{1}{n'^2}]$. We set the distances $d(c_1, z_k) = 1.3$, $d(c_2, z_k) = 1.4$, $d(a_i, z_k) = 1.3 + \frac{1}{10} \cdot \frac{1-\alpha_i}{\alpha_i}$, $d(b_j, z_k) = 1.4 - \frac{1}{10n'} \cdot \frac{1-\alpha_j}{\alpha_j}$ for all $1 \leq i, j, k \leq n'$. Note that for any fixed k, as i increases, $d(a_i, z_k)$ increases, and as j increases, $d(b_j, z_k)$ increases. This ensures that for all i, j, k, the minimum and maximum distances to z_k in state $[\alpha_{ij}, \alpha_{ij} + \frac{1}{n'^2}]$ will always be $d(x_i, z_k)$ and $d(y_j, z_k)$, respectively. The distances are chosen so that the new breakpoints are $\alpha_{ijk} = \frac{i}{n'} + \frac{j}{n'^2} + \frac{k}{n'^3}$.

In phase 4, the decisions are whether to merge $v_1, \ldots, v_{n'}$ to A or C. As in the previous round, we set up the distances so that the distances in the merge equation in state $[\alpha_{ijk}, \alpha_{ijk} + \frac{1}{n'^3}]$ are $d(x_i, v_\ell)$, $d(y_j, v_\ell)$, $d(z_k, v_\ell)$, and $d(c_1, v_\ell)$. We set up the distances so that the breakpoints are exactly $\alpha_{ijk\ell} = \frac{i}{n'} + \frac{j}{n'^2} + \frac{k}{n'^3} + \frac{\ell}{n'^4}$. Finally, in phase 5, the decisions are whether to merge $w_1, \ldots, w_{n'}$ to B or C. The distances in state $[\alpha_{ijk\ell}, \alpha_{ijk\ell} + \frac{1}{n'^4}]$ are $d(x_i, w_m)$, $d(y_j, w_m)$, $d(z_k, w_m)$, and $d(v_\ell, w_m)$. We set the distances so that the final breakpoints are $\alpha_{ijk\ell m} = \frac{i}{n'} + \frac{j}{n'^2} + \frac{k}{n'^3} + \frac{\ell}{n'^4} + \frac{m}{n'^5}$. □

$\Omega(n^7)$ **lower bound** Even with Lemma 3.3.20, there is still a gap between the lower bound of $\Omega(n^5)$ and the upper bound of $O(n^8)$. In this section, we give a sketch for a lower bound of $\Omega(n^7)$. Recall that each breakpoint is an equation in 4 distances, or 8 points. In the $\Omega(n^5)$ proof, the first four rounds can be thought of as a setup to obtain $\Omega(n^4)$ different states, where each state has two sets, $A = \{x_i, y_j\}$ and $B = \{z_k, v_\ell\}$, and then there are $n' = \frac{n}{10}$ more rounds deciding whether to merge $w_1, \ldots, w_{n'}$ to A or B. Therefore, the breakpoint equations depend on points x_i, y_j, z_k, v_ℓ, and w_m. Specifically, the four distances are $d(x_i, w_m)$, $d(y_j, w_m)$, $d(z_k, w_m)$, and $d(v_\ell, w_m)$.

Note that the $\Omega(n^5)$ construction has the nice property that every single merge round uses unique distances, e.g., for all $1 \leq m \leq n'$, round m uses only the distances from all other points to x_m. If we go beyond $\Omega(n^5)$, we will not have this nice property. It is natural to ask whether we can continue this construction into phases 6, 7, and 8, where the equations depend on 6, 7, and 8 points, respectively. It is theoretically possible to extend to two more phases using the following construction. Start the first five phases as in the current proof. Then, in phase 6, we will have sets $A = \{x_i, y_j\}$, $B = \{z_k, v_\ell\}$, and $C = \{w_m\}$. Now define n' new points, $u_1, \ldots, u_{n'}$. For each $1 \leq p \leq n'$, the decision in phase 6 is whether to merge u_p to A, or B to C. The corresponding distances are $d(u_p, x_i)$, $d(u_p, y_j)$, $d(z_k, w_m)$, and $d(v_\ell, w_m)$. Thus, there are six variables which range in $[1, \ldots, n']$, so theoretically this will give n'^6 breakpoints. The challenge is to construct a clustering instance which achieves this setup, with sets A, B, C.

This can theoretically be pushed to one more phase, as well. To achieve $\Omega(n^7)$ breakpoints, we can use a setup with sets $A = \{x_i, y_j\}$, $B = \{z_k, v_\ell\}$, and $C = \{w_m, u_p\}$, and points $t_1, \ldots, t_{n'}$. The decisions are whether to merge t_q to A or B to C, which would have seven variables. Note that it is unclear whether $\Omega(n^8)$ breakpoints are possible. If we use the same format, we would need n' variables $s_1, \ldots, s_{n'}$, and the decision is whether to merge s_r to A or B to C. But whichever merge is chosen, then in the next round, we will not have seven variables, since at least two of the variables must now be contained in the same set, because of the last merge. However, it could be possible to achieve $\Omega(n^8)$ breakpoints with a different type of construction.

3.4 (α, β)-Lloyds++

In this section, we define an infinite family of algorithms generalizing Lloyd's algorithm, with one parameter controlling the the initialization procedure, and another parameter controlling the local search procedure. Our main results bound the intrinsic complexity of this family of algorithms (Theorems 3.4.8 and 3.4.10) and lead to sample complexity results guaranteeing the empirically optimal parameters over a sample are close to the optimal parameters over the unknown distribution. We measure optimality in terms of agreement with the target clustering. We also show theoretically that no parameters are optimal over all clustering applications (Theorem 3.4.4). Finally, we give an efficient algorithm for learning the best initialization parameter (Theorem 3.4.13).

Our family of algorithms is parameterized by choices of $\alpha \in [0, \infty) \cup \{\infty\}$ and $\beta \in [1, \infty) \cup \{\infty\}$. Each choice of (α, β) corresponds to one local search algorithm. A summary of the algorithm is as follows (see Algorithm 9). The algorithm has two phases. The goal of the first phase is to output k initial centers. Each center is iteratively chosen by picking a point with probability proportional to the minimum distance to all centers picked so far, raised to the power of α. The second phase is an iterative two step procedure similar to Lloyd's method, where the first step is to create a Voronoi partitioning of the points induced by the initial set of centers, and then a new set

of centers is chosen by computing the ℓ_β mean of each Voronoi tile.

Algorithm 9 (α, β)-Lloyds++ Clustering

Input: Instance $\mathcal{V} = (V, d, k)$, parameter α.

Phase 1: Choosing initial centers with d^α-sampling

1. Initialize $C = \emptyset$ and draw a vector $\mathbf{Z} = \{z_1, \ldots, z_k\}$ from $[0, 1]^k$ uniformly at random.
2. For each $t = 1, \ldots, k$:
   (a) Partition $[0, 1]$ into n intervals, where there is an interval I_{v_i} for each v_i with size equal to the probability of choosing v_i during d^α-sampling in round t (see Figure 3.10).
   (b) Denote c_t as the point such that $z_t \in I_{c_t}$, and add c_t to C.

Phase 2: Lloyd's algorithm

5. Set $C' = \emptyset$. Let $\{C_1, \ldots, C_k\}$ denote the voronoi tiling of V induced by centers C.
6. Compute $\text{argmin}_{x \in V} \sum_{v \in C_i} d(x, v)^\beta$ for all $1 \leq i \leq k$, and add it to C'.
7. If $C' \neq C$, set $C = C'$ and goto 5.

Output: Centers C and clustering induced by C.

Our goal is to find parameters which return clusterings close to the ground-truth in expectation. Setting $\alpha = \beta = 2$ corresponds to the k-means++ algorithm. The seeding phase is a spectrum between random seeding ($\alpha = 0$), and farthest-first traversal ($\alpha = \infty$), and the Lloyd's algorithm can optimize for common clustering objectives including k-median ($\beta = 1$), k-means ($\beta = 2$), and k-center ($\beta = \infty$).

We start with two structural results about the family of (α, β)-Lloyds++ clustering algorithms. The first shows that for sufficiently large α, phase 1 of Algorithm 9 is equivalent to farthest-first traversal. This means that it is sufficient to consider α parameters in a bounded range.

Farthest-first traversal [Gonzalez, 1985] starts by choosing a random center, and then iteratively choosing the point farthest to all centers chosen so far, until there are k centers. We assume that ties are broken uniformly at random.

Lemma 3.4.1. *Given a clustering instance $\mathcal{V}$ and $\delta > 0$, if $\alpha > \frac{\log\left(\frac{nk}{\delta}\right)}{\log s}$, then d^α-sampling will give the same output as farthest-first traversal with probability $> 1 - \delta$. Here, s denotes the minimum ratio d_1/d_2 between two distances $d_1 > d_2$ in the point set.*

Proof. Given such a clustering instance $\mathcal{V} = (V, d, k)$ and α, first we note that farthest-first traversal and d^α-sampling both start by picking a center uniformly at random from V. Assume both algorithms have chosen initial center v_1, and let $C = \{v_1\}$ denote the set of current centers. In rounds 2 to n, farthest-first traversal deterministically chooses the center u which maximizes $d_{\min}(u, C)$ (breaking ties uniformly at random). We will show that with high probability, in every round, d^α-sampling will also choose the center maximizing $d_{\min}(u, C)$ or break ties at random. In round t, let $d_t = \max_{u \in V} d_{\min}(u, C)$ (assuming C are the first $t - 1$ centers chosen by farthest-first traversal). Let d'_t denote the largest distance smaller than d_t, so by assumption, $d_t > s \cdot d'_t$. Assume there are x points whose minimum distance to C is d_t. Then the probability that d^α-sampling will

fail to choose one of these points is at most

$$\begin{aligned}\frac{(n-x)d_t'^{\alpha}}{(n-x)d_t'^{\alpha}+x(sd_t')^{\alpha}} &\leq \frac{n-x}{n-x+x\cdot s^{\alpha}}\\ &\leq \frac{n}{s^{\alpha}}\end{aligned}$$

Over all k rounds of the algorithm, the probability that d^{α}-sampling will deviate from farthest-first traversal (assuming they start with the same first choice of a center and break ties at random in the same way) is at most $\frac{nk}{s^{\alpha}}$, and if we set this probability smaller than δ and solve for α, we obtain

$$\alpha > \frac{\log\left(\frac{nk}{\delta}\right)}{\log s}.$$

□

For some datasets, $\frac{1}{\log s}$ might be very large. Empirically, it has been shown that (α,β)-Lloyds++ behaves the same as farthest-first traversal for $\alpha > 20$ [Balcan et al., 2018].

Now we define a common stability assumption called separability [Kobren et al., 2017, Pruitt et al., 2011], which states that there exists a value r such that all points in the same cluster have distance less than r and all points in different clusters have distance greater than r.

Definition 3.4.2. *A clustering instance satisfies $(1+c)$-separation if*

$$(1+c)\max_{i|u,v\in C_i} d(u,v) < \min_{j\neq j'|u\in C_j, v\in C_{j'}} d(u,v).$$

Now we show that under $(1+c)$-separation, d^{α}-sampling will give the same output as farthest-first traversal with high probability if $\alpha > \log n$, even for $c = .1$.

Lemma 3.4.3. *Given a clustering instance $\mathcal{V}$ satisfying $(1+c)$-separation, and $0 < \delta$, then if $\alpha > \frac{1}{c}\cdot\left(\log n + \log\frac{1}{\delta}\right)$, with probability $> 1-\delta$, d^{α}-sampling with Lloyd's algorithm will output the optimal clustering.*

Proof. Given $\mathcal{V}$ satisfying $(1+c)$-separation, we know there exists a value r such that for all i, for all $u, v \in C_i$, $d(u,v) < r$, and for all $u \in C_j$, $v \in C_{j'\neq j}$, $d(u,v) > (1+c)r$. WLOG, let $r = 1$. Consider round t of d^{α}-sampling and assume that each center v in the current set of centers C is from a unique cluster in the optimal solution. Now we will bound the probability that the center chosen in round t is not from a unique cluster in the optimal solution. Given a point u from a cluster already represented in C, there must exist $v \in C$ such that $d(u,v) < 1$, so $d_{min}(u,C) < 1$. Given a point u from a new cluster, it must be the case that $d_{min}(u,C) > (1+c)$. The total number of points in represented clusters is $< n$. Then the probability we pick a point from an already represented cluster is $\leq \frac{tn}{tn+c^{\alpha}}$. If we set $\frac{tn}{tn+c^{\alpha}} \leq \frac{\delta}{k}$ and solve for α, we obtain $\alpha > \frac{\log n+\log\frac{1}{\delta}}{\log c} \leq \frac{1}{c}\cdot\left(\log n+\log\frac{1}{\delta}\right)$. Since this is true for an arbitrary round t, and there are k rounds in total, we may union bound over all rounds to show the probability d^{α}-sampling outputting

one center per optimal clustering is $> 1 - \delta$. Then, using $(1 + c)$-separation, the Voronoi tiling of these centers must be the optimal clustering, so Lloyd's algorithm converges to the optimal solution in one step. □

Next, to motivate learning the best parameters, we show that for *any* pair of parameters (α^*, β^*), there exists a clustering instance such that (α^*, β^*)-Lloyds++ outperforms all other values of α, β. This implies that d^β-sampling is not always the best choice of seeding for the ℓ_β objective. Let $\texttt{clus}_{\alpha,\beta}(\mathcal{V})$ denote the expected cost of the clustering outputted by (α, β)-Lloyds++, with respect to the target clustering. Formally, $\texttt{clus}_{\alpha,\beta}(\mathcal{V}) = \mathbb{E}_{\mathbf{Z}\sim[0,1]^k}\left[\texttt{clus}_{\alpha,\beta}\left(\mathcal{V}, \mathbf{Z}\right)\right]$, where $\texttt{clus}_{\alpha,\beta}\left(\mathcal{V}, \mathbf{Z}\right)$ is the cost of the clustering outputted by (α, β)-Lloyds++ with randomness $\mathbf{Z}] \in [0, 1]^k$ (see line 1 of Algorithm 9).

Theorem 3.4.4. *For $\alpha^* \in [0, \infty) \cup \{\infty\}$ and $\beta^* \in [1, \infty) \cup \{\infty\}$, there exists a clustering instance $\mathcal{V}$ whose target clustering is the optimal ℓ_{β^*} clustering, such that $\texttt{clus}_{\alpha^*,\beta^*}(\mathcal{V}) < \texttt{clus}_{\alpha,\beta}(\mathcal{V})$ for all $(\alpha, \beta) \neq (\alpha^*, \beta^*)$.*

Proof. First we give a proof sketch, and later we give the full proof. Consider $\alpha^*, \beta^* \in [0, \infty) \cup \{\infty\}$. The clustering instance consists of 6 clusters, $C_1, \ldots, C_6$. The target clustering will be the optimal ℓ_{β^*} objective. The proof consists of three sections. First, we construct $C_1, \ldots, C_4$ so that d^{α^*} sampling has the best chance of putting exactly one point into each optimal cluster. Then we add "local minima traps" to each cluster, so that if any cluster received two centers in the sampling phase, Lloyd's method will not be able to move the centers to a different cluster. Finally, we construct C_5 and C_6 so that if seeding put one point in each cluster, then β^*-Lloyd's method will outperform any other $\beta \neq \beta^*$.

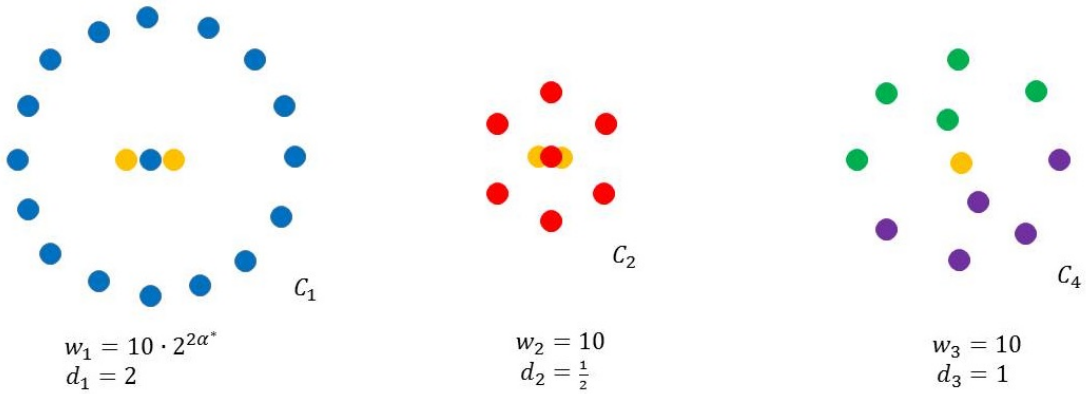


Figure 3.9: Optimal instance for d^{α^*}-sampling

Step 1: we let C_1 and C_2 equal two different cliques, and we define a third clique whose points are equal to $C_3 \cup C_4$. See Figure 3.9. We space these cliques arbitrarily far apart so that with high probability, the first three sampled centers will each be in a different clique. Now the idea is to define the distances and sizes of the cliques so that α^* is the value of α with the greatest chance of putting the last center into the third clique. If we set the distances in cliques 1,2,3 to 2, $1/2$, and 1, and set $|C_1| = 2^{2\alpha^*}|C_2|$, then the probability of sampling a 4th center in the third clique for

$\alpha = \alpha^* + \delta$ is equal to

$$\frac{|C_3 \cup C_4|}{|C_3 \cup C_4| + (2^{\alpha^*+\delta} + 2^{\alpha^*-\delta})|C_2|}.$$

This is maximized when $\delta = 0$.

Now we add local minima traps for Lloyd's method as follows. In the first two cliques, we add three centers so that the 2-clustering cost is only slightly better than the 1-clustering cost. In the third clique, which consists of $C_3 \cup C_4$, add centers so that the 2-clustering cost is much lower than the 1-clustering cost. We also show that since all cliques are far apart, it is not possible for a center to move between clusters during Lloyd's method.

Finally, we add three centers c_5, b_5, b_5' to the last cluster C_5. We set the rest of the points so that c_5 minimizes the ℓ_{β^*} objective, while b_5 and b_5' favor $\beta = \beta^* \pm \epsilon$. Therefore, (α^*, β^*) performs the best out of all pairs (α, β).

Now we give the full details of the proof. Consider $\alpha^*, \beta^* \in [0, \infty) \cup \{\infty\}$. The clustering instance consists of 6 clusters, $C_1, \ldots, C_6$. The target clustering will be the optimal ℓ_{β^*} objective. The basic idea of the proof is as follows.

First, we show that for all β and $\alpha \neq \alpha^*$, $\texttt{clus}_{\alpha^*,\beta}(\mathcal{V}) < \texttt{clus}_{\alpha,\beta}(\mathcal{V})$. We use clusters $C_1, \ldots, C_4$ to accomplish this. We set up the distances so that d^{α^*} sampling is more likely to sample one point per cluster than any other value of α. If the sampling does not sample one point per cluster, then it will fall into a high-error local minima trap that β-Lloyd's method cannot escape, for any value of β. Therefore, d^{α^*} sampling is more effective than any other value of α.

Next, we use clusters C_5 and C_6 to show that if we start with one center in C_5 and one in C_6, then β^*-Lloyd's method will strictly outperform any other value of β. We accomplish this by adding three choices of centers for C_5. Running β^*-Lloyd's method will return the correct center, but any other value of β will return suboptimal centers which incur error on C_5 and C_6. Also, we show that Lloyd's method returns the same centers on $C_1, \ldots, C_4$, independent of β.

For the first part of the proof, we define three cliques (see Figure 3.9). The first two cliques are C_1 and C_2, and the third clique is $C_3 \cup C_4$. C_1 contains w_1 points at distance $x > 1$, and C_2 contains w_2 points at distance $\frac{1}{x}$. We set $w_1 = x^{2\alpha^*} w_2$. The last clique contains w points at distance 1. Since the cliques are very far apart, the first three sampled centers will each be in a different clique, with high probability (for $\alpha > .1$). The probability of sampling a 4th center x_4 in the third clique, for $\alpha = \alpha^* + \delta$ is equal to

$$\frac{w}{w + (x^{\alpha^*+\delta} + x^{\alpha^*-\delta})w_2}.$$

Since $x^{\alpha^*+\delta} + x^{\alpha^*-\delta}$ is minimized when $\delta = 0$, this probability is maximized when $\alpha = \alpha^*$. Now we show that the error will be much larger when x_4 is not in the third clique. We add center c_1 which is distance $x - \epsilon$ to all points in C_1. We also add centers b_1 and b_1' which are distance $x - 2\epsilon$ to B_1 and B_1' such that B_1 and B_1' form a partition of C_1. Similarly, we add centers c_2, b_2, and b_2' at distance $\frac{1}{x} - \epsilon$ and $\frac{1}{x} - 2\epsilon$ to C_2, B_2, and B_2', respectively, such that B_2 and B_2' form a partition of C_2. Finally, we add c_3 and c_4 which are distance .5 to C_3 and C_4, respectively, and we add b_3 which is distance $1 - \epsilon$ to $C_3 \cup C_4$. Then the optimal centers for any β must be c_1, c_2, c_3, c_4, and this will be the solution of β-Lloyd's method, as long as the sampling procedure returned one

point in the first two cliques, and two points in the third clique. If the sampling procedure returns two points in the first clique or second clique, then β-Lloyd's method will return b_1, b_1', c_2, b_3 or c_1, b_2, b_2', b_3, respectively. This will incur error $w/2 + w_1/2$ or $w/2 + w_2/2$, since we set the target clustering to be the optimal ℓ_{β^*} objective which is equal to $\{C_1, C_2, C_3, C_4\}$. Note that we have set up the distances so that Lloyd's method is independent of β. Therefore, the expected error is equal to

$$\frac{w}{w + (x^{\alpha^*+\delta} + x^{\alpha^*-\delta})w_2} \cdot (w/2 + \min(w_1, w_2)/2).$$

This finishes off the first part of the proof. Next, we construct C_5 and C_6 so that β^*-Lloyd's method will return the best solution, assuming the sampling returned one point in C_5 and C_6. Later we will show how adding these clusters does not affect the previous part of the proof. We again define three cliques. The first clique is size $\frac{2}{3}w_5$ and distance .1, the second clique is size $\frac{1}{3}w_5$ and distance .1, and the third clique is size w_6 and distance .1. The first two cliques are C_5, and the second clique is C_6. The distance between the first two cliques is .2, and the distance between the first two and the third clique is 1000. Now imagine the first two cliques are parallel to each other, and there is a perpendicular bisector which contains possible centers for C_5. I.e., we will consider possible centers c for C_5 where the ℓ_β cost of c is $\frac{2}{3}w_5 \cdot z^\beta + \frac{1}{3}w_5(.2-z)^\beta$ for some $0 \leq z \leq .2$, . For $\beta \in (0, \infty)$, the β which minimizes the expression must be in $[0, .2]$. We set c_5 corresponding to the z which minimizes the expression for β^*, call it z^*. Therefore, β^*-Lloyd's method will output c_5. We also set centers b_5 and b_5' corresponding to $z^* - \epsilon$ and $z^* + \epsilon$. Therefore, any value of β slightly above or below β^* will return a different center. We add a center C_6 at distance $.1 - \epsilon$ to the third clique. This is the only point we add, so it will always be chosen by β-Lloyd's method for all β. Finally, we add two points p_1 and p_2 in between the second and third cliques. We set the distances as follows. $d(c_5, p_1) = d(c_6, p_2) = 500 - \epsilon$, $d(c_5, p_2) = d(c_6, p_1) = 500$, $d(b_5, p_1) = 500 + \epsilon$, and $d(b_5', p_2) = 500 - 2\epsilon$. Since the weight of these two points are very small compared to the cliques, these points will have no effect on the prior sampling and Lloyd's method analyses. The optimal clustering for the ℓ_{β^*} objective is to add p_1 to C_5 and p_2 to C_6. However, running β-Lloyd's method for β smaller or larger than β^* will return center b_5 or b_5' and incur error 1 by mislabeling p_1 or p_2.

Since all cliques from both parts of the proof are 1000 apart, with high probability, the first 5 sampled points will be in cliques $C_1, C_2, C_3 \cup C_4, C_5$, and C_6. Since the cliques from the second part of the proof are distance .2, while in the first part they were $> \frac{1}{x}$ apart, we can set the variables x, w_1, w_2, w, w_5, w_6 so that with high probability, the sixth sampled point will not be in C_5 or C_6. Therefore, the constructions in the second part do not affect the first part. This concludes the proof. □

3.4.1 Sample efficiency

Now we give sample complexity bounds for learning the best algorithm from the (α, β)-Lloyds++ family. We analyze the phases of Algorithm 9 separately. For the first phase, our main structural result is to show that for a given clustering instance and value of β, with high probability over the randomness in Algorithm 9, the number of discontinuities of the cost function $\texttt{clus}_{\alpha,\beta}(\mathcal{V}, \mathbf{Z})$ as we vary $\alpha \in [0, \alpha_h]$ is $O(nk(\log n)\alpha_h)$. Our analysis crucially harnesses the randomness in the algorithm to achieve this bound. For instance, if we use a combinatorial approach as in prior

algorithm configuration work, we would only achieve a bound of $n^{O(k)}$, which is the total number of sets of k centers. For completeness, we give a combinatorial proof of $O(n^{k+3})$ discontinuities before we show the stronger result.

Combinatorial upper bound We upper bound the number of discontinuities of $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$. Recall that $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ denotes the outputted centers from phase 1 of Algorithm 9 on instance $\mathcal{V}$ with randomness $\mathbf{Z}$. For the first phase, our main structural result is to show that for a given clustering instance and value of β, with high probability over the randomness in Algorithm 9, the number of discontinuities of the cost function $\texttt{clus}_{\alpha,\beta}(\mathcal{V}, \mathbf{Z})$ as we vary $\alpha \in [0, \alpha_h]$ is $O(nk(\log n)\alpha_h)$. Our analysis crucially harnesses the randomness in the algorithm to achieve this bound. In contrast, a combinatorial approach would only achieve a bound of $n^{O(k)}$, which is the total number of sets of k centers. For completeness, we start with a combinatorial proof of $O(n^{k+3})$ discontinuities. Although Theorem 3.4.5 is exponential as opposed to Theorem 3.4.8, it holds with probability 1 and has no dependence on α_h.

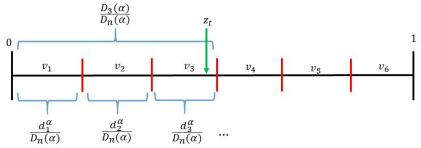
Theorem 3.4.5. *Given a clustering instance $\mathcal{V}$ and vector $\mathbf{Z} \in [0,1]^k$, the number of discontinuities of $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ as a function of α over $[0, \infty) \cup \{\infty\}$ is $O\left(\min\left(n^{k+3}, n^2 2^n\right)\right)$.*

Proof. Given a clustering instance $\mathcal{V}$ and a vector $\mathbf{Z}$, consider round t of the d^α seeding algorithm. At this point, there are $\binom{n}{t-1}$ choices for the set of current centers C. Recall that the decision for the next center depends on $z_t \in [0,1]$. We denote the points $V = \{v_1, \ldots, v_n\}$ and WLOG assume the algorithm orders the intervals $I_{v_1}, I_{v_2}, \ldots, I_{v_n}$. Given a point $v_i \in V$, it will be chosen as the next center if and only if z_t lands in its interval, formally,

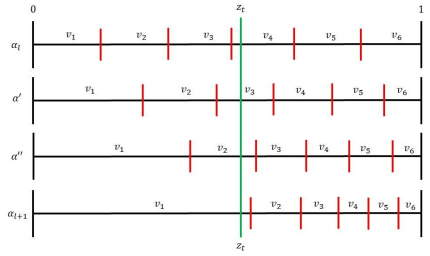
$$\frac{\sum_{j=1}^{i-1} d_{\min}(v_j, C)^\alpha}{\sum_{j=1}^{n} d_{\min}(v_j, C)^\alpha} < z_t < \frac{\sum_{j=1}^{i} d_{\min}(v_j, C)^\alpha}{\sum_{j=1}^{n} d_{\min}(v_j, C)^\alpha}$$

By a consequence of Rolle's theorem, these two equations have at most $n+1$ roots each. Therefore, in the $n+2$ intervals of α between each root, the decision whether or not to choose v_i as a center in round t is fixed. Note that the center set C, the point v_i, and the number z_t fixed the coefficients of the equation. Since in round t, there are $\binom{n}{t}$ choices of centers, then there are $\sum_{t=1}^{k}\binom{n}{k} \cdot n \cdot 2$ total equations which determine the outcome of the algorithm. Each equation has at most $n+1$ roots, so it follows there are $1 + \sum_{t=1}^{k}\binom{n}{k} \cdot n \cdot 2(n+1) \in O(n^3 \cdot n^k)$ total intervals of α along $[0, \infty) \cup \{\infty\}$ such that within each interval, the entire outcome of the algorithm, $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$, is fixed. Note that we used $k \cdot n^k$ to bound the total number of choices for the set of current centers. We can also bound this quantity by 2^n, since each point is either a center or not a center. This results in a final bound of $O\left(\min\left(n^{k+3}, n^2 2^n\right)\right)$. □

Probabilistic upper bound To show the $O(nk(\log n)\alpha_h)$ upper bound, we start by giving a few definitions of concepts used in the proof. Assume we start to run Algorithm 9 without a specific setting of α, but rather a range $[\alpha_\ell, \alpha_h]$, for some instance $\mathcal{V}$ and randomness $\mathbf{Z}$. In some round t, if Algorithm 9 would choose a center c_t for every setting of $\alpha \in [\alpha_\ell, \alpha_h]$, then we continue normally. However, if the algorithm would choose a different center depending on the specific value of α used from the interval $[\alpha_\ell, \alpha_h]$, then we fork the algorithm, making one copy for each possible center. In particular, we partition $[\alpha_\ell, \alpha_h]$ into a finite number of sub-intervals such that the next



(a) The algorithm chooses v_3 as a center.



(b) In the interval $[\alpha_\ell, \alpha_{\ell+1}]$, the algorithm may choose v_4, v_3, v_2, or v_1 as a center, based on the value of α.

Figure 3.10: Examples of which centers the algorithm chooses in a given round.

center is constant on each interval. The boundaries between these intervals are "breakpoints", since as α crosses those values, the next center chosen by the algorithm changes. Our goal is to bound the total number of breakpoints over all k rounds in phase 1 of Algorithm 9, which bounds the number of discontinuities of the cost of the outputted clustering as a function of α over $[\alpha_\ell, \alpha_h]$.

A crucial step in the above approach is determining when to fork and where the breakpoints are located. Recall that in round t of Algorithm 9, each datapoint v_i has an interval in $[0, 1]$ of size $\frac{d_i^\alpha}{D_n(\alpha)}$, where d_i is the minimum distance from v_i to the current set of centers, and $D_j(\alpha) = d_1^\alpha + \cdots + d_j^\alpha$. Furthermore, the interval is located between $\frac{D_{i-1}(\alpha)}{D_n(\alpha)}$ and $\frac{D_i(\alpha)}{D_n(\alpha)}$ (see Figure 3.10). WLOG, we assume $d_1 \geq \cdots \geq d_n$. We prove the following nice structure about these intervals.

Lemma 3.4.6. *Assume that $v_1, \ldots, v_n$ are sorted in decreasing distance from a set C of centers. Then for each $i = 1, \ldots, n$, the function $\alpha \mapsto \frac{D_i(\alpha)}{D_n(\alpha)}$ is monotone increasing and continuous along $[0, \infty)$. Furthermore, for all $1 \leq i \leq j \leq n$ and $\alpha \in [0, \infty)$, we have $\frac{D_i(\alpha)}{D_n(\alpha)} \leq \frac{D_j(\alpha)}{D_n(\alpha)}$.*

Proof. Recall that $D_i(\alpha) = \sum_{j=1}^{i} d_{\min}(v_{(i)}, C)^\alpha$, where $v_1, \ldots, v_n$ are the points sorted in decreasing order of distance to the set of centers C.

We prove that for each i, the function $\alpha \mapsto D_i(\alpha)/D_n(\alpha)$ is monotone increasing. Given $\alpha_1 < \alpha_2$, we must show that for each i,

$$\frac{D_i(\alpha_1)}{D_n(\alpha_1)} \leq \frac{D_i(\alpha_2)}{D_n(\alpha_2)}.$$

This is equivalent to showing

$$D_i(\alpha_1) D_n(\alpha_2) \leq D_i(\alpha_2) D_n(\alpha_1).$$

Using the shorthand notation $d_j = d(v_j, C)$, we have

$$\begin{aligned}
D_i(\alpha_1)D_n(\alpha_2) &= \left(\sum_{j=1}^{i} d_j^{\alpha_1}\right)\left(\sum_{j=1}^{n} d_j^{\alpha_2}\right) \\
&= \left(\sum_{j=1}^{i} d_j^{\alpha_1}\right)\left(\sum_{j=1}^{i} d_j^{\alpha_2}\right) + \left(\sum_{j=1}^{i} d_j^{\alpha_1}\right)\left(\sum_{j=i+1}^{n} d_j^{\alpha_2}\right) \\
&= \left(\sum_{j=1}^{i} d_j^{\alpha_1}\right)\left(\sum_{j=1}^{i} d_j^{\alpha_2}\right) + \sum_{j=1}^{i}\sum_{k=i+1}^{n} d_j^{\alpha_1} d_k^{\alpha_2} \\
&\leq \left(\sum_{j=1}^{i} d_j^{\alpha_1}\right)\left(\sum_{j=1}^{i} d_j^{\alpha_2}\right) + \sum_{j=1}^{i}\sum_{k=i+1}^{n} d_j^{\alpha_1} d_k^{\alpha_2}\left(\frac{d_j}{d_k}\right)^{\alpha_2-\alpha_1} \\
&= \left(\sum_{j=1}^{i} d_j^{\alpha_1}\right)\left(\sum_{j=1}^{i} d_j^{\alpha_2}\right) + \sum_{j=1}^{i}\sum_{k=i+1}^{n} d_j^{\alpha_2} d_k^{\alpha_1} \\
&= \left(\sum_{j=1}^{i} d_j^{\alpha_2}\right)\left(\sum_{j=1}^{n} d_j^{\alpha_1}\right) \\
&= D_i(\alpha_2)D_n(\alpha_1),
\end{aligned}$$

as required.

Next we show that $\alpha \mapsto D_i(\alpha)/D_n(\alpha)$ is continuous along $[0, \infty)$. $D_i(\alpha)$ and $D_n(\alpha)$ are both sums of simple exponential functions, so they are continuous. $D_n(\alpha)$ is always at least n along $[0, \infty)$, therefore, $D_i(\alpha)/D_n(\alpha)$ is continuous.

Finally, we show that for all $1 \leq i \leq j \leq n$, we have $\frac{D_i(\alpha)}{D_n(\alpha)} \leq \frac{D_j(\alpha)}{D_n(\alpha)}$. Given $1 \leq i \leq j \leq n$, then

$$\frac{D_j(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha)}{D_n(\alpha)} = \frac{d_{i+1}^{\alpha} + \cdots + d_j^{\alpha}}{D_n(\alpha)} \geq 0$$

This completes the proof. □

This lemma guarantees two crucial properties. First, we know that for every (ordered) set C of $t \leq k$ centers chosen by phase 1 of Algorithm 9 up to round t, there is a single interval (as opposed to a more complicated set) of α-parameters that would give rise to C. Second, for an interval $[\alpha_\ell, \alpha_h]$, the set of possible next centers is exactly $v_{(i_\ell)}, v_{(i_\ell+1)}, \ldots, v_{(i_h)}$, where i_ℓ and i_h are the centers sampled when α is α_ℓ and α_h, respectively (see Figure 3.10). Now we are ready to prove our main structural result. Formally, we define $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ as the outputted centers from phase 1 of Algorithm 9 on instance $\mathcal{V}$ with randomness $\mathbf{Z}$.

Now before we give the proof of Theorem 3.4.8, we need to bound the derivative of $\left(\frac{D_i(\alpha)}{D_n(\alpha)}\right)$.

Lemma 3.4.7. *Given $d_1 \geq \cdots \geq d_n$, and $i = 1, \ldots, n$, for $\alpha \in [0, \alpha_h]$, $\left|\frac{\partial}{\partial\alpha}\left(\frac{D_i(\alpha)}{D_n(\alpha)}\right)\right| \leq 4 \log n$.*

Proof.

$$\begin{aligned}
\frac{\partial}{\partial\alpha}\left(\frac{D_i(\alpha)}{D_n(\alpha)}\right) &\le \frac{(D_i(\alpha)'(D_n(\alpha)) - (D_n(\alpha))'(D_i(\alpha))}{(D_n(\alpha))^2} \\
&\le \frac{\left(\sum_{x=1}^{i} d_x^\alpha \log(d_x^\alpha)\right)\left(\sum_{y=1}^{n} d_y^\alpha\right) - \left(\sum_{x=1}^{i} d_x^\alpha\right)\left(\sum_{y=1}^{n} d_y^\alpha \log(d_y^\alpha)\right)}{\sum_{x=1}^{i}\sum_{y=1}^{i} d_x^\alpha d_y^\alpha} \\
&\le 2 \cdot \frac{\sum_{x=1}^{i}\sum_{y=i+1}^{n} d_x^\alpha d_y^\alpha \left(\log(d_x^\alpha) - \log(d_y^\alpha)\right)}{\sum_{x=1}^{n}\sum_{y=1}^{n} d_x^\alpha d_y^\alpha} \\
&\le 2 \cdot \frac{\sum_{x=1}^{i}\left(d_x^\alpha\left(\sum_{y=i+1}^{n} d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right)\right)\right)}{\sum_{x=1}^{n}\left(d_x^\alpha\left(\sum_{y=i+1}^{n} d_y^\alpha\right)\right)}
\end{aligned}$$

Now we will show that $\sum_{y=i+1}^{n} d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \le 2\log n \sum_{y=1}^{n} d_y^\alpha$ by grouping each $i+1 \le y \le n$ into one of two different cases.

Case 1: $d_y^\alpha > \frac{d_1^\alpha}{n^2}$. Then $d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \le 2 d_y^\alpha \log n$.

Case 2: $d_y^\alpha < \frac{d_1^\alpha}{n^2}$. Then $d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \le d_1^\alpha \left(\frac{d_y^\alpha}{d_1^\alpha}\log\frac{d_1^\alpha}{d_y^\alpha}\right) \le \frac{1}{n}\cdot d_1^\alpha$, where the last inequality follows because $\frac{1}{x}\log x \le \frac{1}{n}$ for all $x > n^2$. Therefore, the sum over all y in case 2 are smaller than d_1^α.

So, for all y in case 1, $d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \le 2 d_y^\alpha \log n$, and $\sum_{y:\text{case 2}} d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \le d_1^\alpha$. It follows that for all $i \ge 1$, $\sum_{y=i+1}^{n} d_y \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \le 2\log n \sum_{y=1}^{n} d_y$. Therefore, $\frac{\partial}{\partial\alpha}\left(\frac{D_i(\alpha)}{D_n(\alpha)}\right) \le 4\log n$. □

Theorem 3.4.8. *Given a clustering instance $\mathcal{V}$, the expected number of discontinuities of the function $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ for α over $[0, \alpha_h]$ is $O(nk(\log n)\alpha_h)$. Here, the expectation is over the uniformly random draw of $\mathbf{Z} \in [0,1]^k$.*

Proof. Given $\mathcal{V}$ and $[0, \alpha_h]$, we will show that $\mathbb{E}[\#I] \le nk\log n \cdot \alpha_h$, where $\#I$ denotes the total number of discontinuities of $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ and the expectation is over the randomness $Z \in [0,1]^k$ of the d^α-sampling algorithm. Consider round t of a run of the algorithm. Suppose at the beginning of round t, there are L possible states of the algorithm, e.g., L sets of α such that within a set, the choice of the first $t-1$ centers is fixed. By Lemma 3.4.6, we can write these sets as $[\alpha_0, \alpha_1], \ldots, [\alpha_{L-1}, \alpha_L]$, where $0 = \alpha_0 < \cdots < \alpha_L = \alpha_h$. Given one interval, $[\alpha_\ell, \alpha_{\ell+1}]$, we claim the expected number of new breakpoints $\#I_{t,\ell}$ by choosing a center in round t is bounded by $4n\log n(\alpha_{\ell+1} - \alpha_\ell)$. Note that $\#I_{t,\ell} + 1$ is the number of possible choices for the next center in round t using α in $[\alpha_\ell, \alpha_{\ell+1}]$.

The claim gives an upper bound on the expected number of new breakpoints, where the expectation is only over z_t (the uniformly random draw from $[0,1]$ used by Algorithm 9 in round t), and the bound holds for any given configuration of $d_1 \ge \cdots \ge d_n$. Assuming the claim, we can finish off the proof by using linearity of expectation as follows. Let $\#I$ denote the total number of discontinuities of $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$.

$$
\begin{aligned}
E_{Z\in[0,1]^k}[\#I] &\le E_{Z\in[0,1]^k}\left[\sum_{t=1}^{k}\sum_{\ell=1}^{L}(\#I_{t,\ell})\right] \\
&\le \sum_{t=1}^{k}\sum_{\ell=1}^{L} E_{Z\in[0,1]^k}[\#I_{t,\ell}] \\
&\le \sum_{t=1}^{k}\sum_{\ell=1}^{L} 4n\log n(\alpha_{\ell+1}-\alpha_\ell) \\
&\le 4nk\log n\cdot\alpha_h
\end{aligned}
$$

Now we will prove the claim. Recall that for α-sampling, each point x receives an interval along $[0,1]$ of size $\frac{d_x^\alpha}{D_n(\alpha)}$, so the number of breakpoints in round t along $[\alpha_\ell,\alpha_{\ell+1}]$ corresponds to the number of times z_t switches intervals as we increase α from α_ℓ to $\alpha_{\ell+1}$. By Lemma 3.4.6, the endpoints of these intervals are monotone increasing, so the number of breakpoints is exactly $x-y$, where x and y are the minimum indices s.t. $\frac{D_x(\alpha_\ell)}{D_n(\alpha_\ell)} > z_t$ and $\frac{D_y(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} > z_t$, respectively. We want to compute the expected value of $x-y$ for z_t uniform in $[0,1]$ (here, x and y are functions of z_t).

We take the approach of analyzing each interval individually. One method for bounding $E_{z_t\in[0,1]}[x-y]$ is to compute the maximum possible number of breakpoints for each interval I_{v_j}, for all $1\le j\le n$. Specifically, if we let i denote the minimum index such that $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} < \frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then

$$
\begin{aligned}
E[\#I_{t,\ell}] &\le \sum_{j=1}^{n} P\left(\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} < z_t < \frac{D_{j+1}(\alpha_\ell)}{D_n(\alpha_\ell)}\right)\cdot(j-i+1) \\
&\le \sum_{j=1}^{n} \frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)}\cdot(j-i+1).
\end{aligned}
$$

In this expression, we are using the worst case number of breakpoints within each bucket, $j-i+1$.

We cannot quite use this expression to obtain our bound; for example, when $\alpha_{\ell+1}-\alpha_\ell$ is extremely small, $j-i+1=1$, so this expression will give us $E[\#I_{t,\ell}]\le 1$ over $[\alpha_\ell,\alpha_{\ell+1}]$, which is not sufficient to prove the claim. Therefore, we give a more refined analysis by further breaking into cases based on whether z_t is smaller or larger than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$. If z_t is less than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then we have the maximum number of breakpoints possible, since the algorithm chooses center v_{i-1} when $\alpha=\alpha_{\ell+1}$ and it chooses center v_j when $\alpha=\alpha_\ell$. The number of breakpoints is therefore $j-i+1$, by Lemma 3.4.6. We denote this event by $E_{t,j}$, i.e., $E_{t,j}$ is the event that in round t, z_t lands in I_{v_j} and is less than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$. If z_t is instead greater than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then the algorithm chooses center v_i when $\alpha=\alpha_{\ell+1}$ (or another center $v_{i'}$ where $i'>i$), so the number of breakpoints is $\le j-i$. We denote this event by $E'_{t,j}$. See Figure 3.11. Note that $E_{t,j}$ and $E'_{t,j}$ are disjoint and $E_{t,j}\cup E'_{t,j}$ is the event that $z_t\in I_{v_j}$.

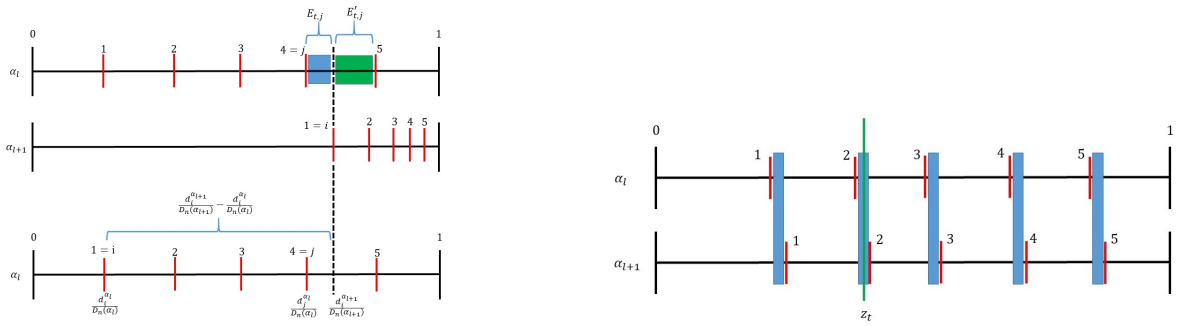


Figure 3.11: Definition of $E_{t,j}$ and $E'_{t,j}$, and details for bounding $j-i$ (left). Intuition for bounding $P(E_{t,j})$, where the blue regions represent $E_{t,j}$ (right).

Within an interval I_{v_j}, the expected number of breakpoints is

$$P(E_{t,j})(j-i+1) + P(E'_{t,j})(j-i) = P(E_{t,j} \cup E_{t,j})(j-i) + P(E'_{t,j}).$$

We will show that $j-i$ and $P(E_{t,j})$ are both proportional to $(\log n)(\alpha_{\ell+1} - \alpha_\ell)$, which finishes off the claim.

First we upper bound $P(E_{t,j})$. Recall this is the probability that z_t is in between $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$ and $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, which is

$$\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} \leq \frac{D_j(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}.$$

Therefore, we can bound this quantity by bounding the derivative $\left|\frac{\partial}{\partial \alpha}\left(\frac{D_j(\alpha)}{D_n(\alpha)}\right)\right|$, which is at most $4 \log n$ by Lemma 3.4.7.

For case 1, recall that $j-i$ represents the number of intervals between $\frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}$ and $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$. Note that the smallest interval in this range is $\frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)}$, and $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)} \leq \frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}$. Therefore, the expected number of breakpoints is at most $\frac{D_n(\alpha_\ell)}{d_j^{\alpha_\ell}} \cdot \left(\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}\right)$, and we can bound the second half of this fraction by again using Lemma 3.4.7.

Putting case 1 and case 2 together, we have

$$
\begin{aligned}
E[\#I_{t,\ell}] &\le \sum_j \big(P(E'_{t,j}) \cdot (j-i) + P(E_{t,j}) \cdot (j-i+1)\big) \\
&\le \sum_j \big(P(E'_{t,j}) \cdot (j-i) + P(E_{t,j}) \cdot (j-i) + P(E_{t,j})\big) \\
&\le \sum_j \big(P(E'_{t,j} \cup E_{t,j}) \cdot (j-i) + P(E_{t,j})\big) \\
&\le \sum_j \big(P(z_t \in I_{v_j}) \cdot (j-i)\big) + \sum_j P(E_{t,j}) \\
&\le \sum_j \left(\frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)}\right)\left(\frac{D_n(\alpha_\ell)}{d_j^{\alpha_\ell}} \cdot 4\log n(\alpha_{\ell+1}-\alpha_\ell)\right) + \sum_j (4\log n(\alpha_{\ell+1}-\alpha_\ell)) \\
&\le \sum_j (4\log n(\alpha_{\ell+1}-\alpha_\ell)) + \sum_j (4\log n(\alpha_{\ell+1}-\alpha_\ell)) \\
&\le 8n\log n(\alpha_{\ell+1}-\alpha_\ell)
\end{aligned}
$$

This concludes the proof. □

In fact, we can also show that the worst-case number of discontinuities is exponential.

Lemma 3.4.9. *Given n, there exists a clustering instance $\mathcal{V}$ of size n and a vector $\mathbf{Z}$ such that the number of discontinuities of $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ as a function of α over $[0, 2]$ is $2^{\Omega(n)}$.*

Proof. We construct $\mathcal{V} = (V, d, k)$ and $\mathbf{Z} = \{z_1, \ldots, z_k\}$ such that $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ has $2^{k/3}$ different intervals in α which give different outputs.

Here is the outline of the construction. At the start, we set z_1 so that one point, v, will always be the first center chosen. Then we add points $a_1, b_1, \ldots, a_k, b_k$ such that in round i, either a_i or b_i will be chosen as centers. We carefully set the distances so that for each combinations of centers, there is an α interval which achieves this combination of centers. Therefore, the total number of α intervals such that the output of the sampling step is fixed, is 2^{k-1}. Our construction also uses points $a'_1, b'_1, \ldots a'_k, b'_k$ and $v_1, \ldots, v_k$ which are never chosen as centers, but will be crucial in the analysis.

Next, we describe the distances between the points in our clustering instance. Almost all distances will be set to 100, except for a few distances: for all i, $d(a_i, a'_i) = d(b_i, b'_i) = \epsilon$, $d(b_i, v_i) = 100 - o_i$, $d(a_i, b_i) = 2o_i$, $d(v, a_1) = d(v, b_1) = 99$, and $d(a_{i-1}, a_i) = d(a_{i-1}, b_i) = d(b_{i-1}, a_i) = d(b_{i-1}, b_i) = 100 - o_i$, for $0 \le \epsilon, o_1, \ldots, o_k \le 1$ to be specified later. At the end, we will perturb all other distances by a slight amount ($< \epsilon$) away from 100, to break ties.

Now we set up notation to be used in the remainder of the proof. We set $z_i = \frac{1}{2}$ for all i. For $1 \le i \le k$, given $\mathbf{x} \in \{0, 1\}^{i-1}$, let $E_{\mathbf{x}}$ denote the equation in round i which determines whether a_i or b_i is chosen as the next center, in the case where for all $1 \le j < i$, $a_j \in C$ if $\mathbf{x}_j = 0$, or else $b_j \in C$ (and let E' denote the single equation in round 2). Specifically, $E_{\mathbf{x}}$ is the following expression

$$\frac{100^\alpha \left(\frac{n-(i-1)}{2}\right)}{100^\alpha(n-2(i-1)) + (100-o_i)^\alpha + \sum_{j=1}^{i-1}(100-\mathbf{x}_j o_j)^\alpha}.$$

Let $\alpha_{\mathbf{x}}$ denote the solution to equation $E_{\mathbf{x}} = \frac{1}{2}$ in $[1, 3]$, if it exists. In the rest of the proof, we must show there exist parameters $\epsilon, o_0, \ldots, o_k$ which admit an ordering to the values $\alpha_{\mathbf{x}}$ which ensures that each $\alpha_{\mathbf{x}}$ falls in the correct range to split up each interval, thus achieving 2^{k-1} intervals. The ordering of the $\alpha_{\mathbf{x}}$'s can be specified by two conditions: *(1)* $\alpha_{[\mathbf{x}\ 0]} < \alpha_{[\mathbf{x}]} < \alpha_{[\mathbf{x}\ 1]}$ and *(2)* $\alpha_{[\mathbf{x}\ 0\ \mathbf{y}]} < \alpha_{[\mathbf{x}\ 1\ \mathbf{z}]}$ for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \bigcup_{i<k}\{0,1\}^i$ and $|\mathbf{y}| = |\mathbf{z}|$. To prove the $\alpha_{\mathbf{x}}$'s follow this ordering, we use an inductive argument. We must show the following claim: there exist $0 < o_1, \ldots, o_k < 1$ such that if we solve $E_{\mathbf{x}} = \frac{1}{2}$ for $\alpha_{\mathbf{x}}$ for all $\mathbf{x} \in \cup_{i<k}\{0,1\}^i$, then the α's satisfy $\alpha_{[\mathbf{x}\ 0]} < \alpha_{[\mathbf{x}]} < \alpha_{[\mathbf{x}\ 1]}$ and for all $i < k$, $\alpha_{[\mathbf{x}\ 1]} < \alpha_{[\mathbf{y}\ 0]}$ for $\mathbf{x}, \mathbf{y} \in \{0,1\}^i$ and $x_1 \ldots x_i < y_1 \ldots y_i$.

Given $\mathbf{x} \in \{0,1\}^i$, for $1 \le i \le k-1$, let $p(\mathbf{x}), n(\mathbf{x}) \in \{0,1\}^i$ denote the vectors which sit on either side of $\alpha_{\mathbf{x}}$ in the desired ordering, i.e., $\alpha_{\mathbf{x}}$ is the only $\alpha_{\mathbf{y}}$ in the range $(\alpha_{p(\mathbf{x})}, \alpha_{n(\mathbf{x})})$ such that $|\mathbf{y}| = i$. If $\mathbf{x} = [1 \ldots 1]$, then set $\alpha_{n(\mathbf{x})} = 3$, and if $\mathbf{x} = [0 \ldots 0]$, then set $\alpha_{p(\mathbf{x})} = 1$.

Given $1 \le i \le k-2$, assume there exist $0 < o_1, \ldots, o_i < 1$ such that the statement is true. Now we will show the statement holds for $i+1$. Given $\mathbf{x} \in \{0,1\}^i$, by assumption, we have that the solution to $E_{\mathbf{x}} = \frac{1}{2}$ is equal to $\alpha_{\mathbf{x}}$. First we consider $E_{[\mathbf{x}\ 0]} = \frac{1}{2}$. Note there are two differences in the equations $E_{\mathbf{x}}$ and $E_{[\mathbf{x}\ 0]}$. First, the number of 100^α terms in the numerator decreases by 1, and the number of terms in the denominator decreases by 2. WLOG, at the end we set a constant $c = \frac{n}{k}$ large enough so that this effect on the root of the equation is negligible for all n. Next, the offset in the denominator changes from $(100-o_i)^\alpha$ to $(100-o_{i+1})^\alpha$. Therefore, if $0 < o_{i+1} < o_i$, then $\alpha_{[\mathbf{x}\ 0]} < \alpha_{\mathbf{x}}$. Furthermore, there exists an upper bound $0 < z_{i+1} < o_i$ such that for all $0 < o_{i+1} < z_{i+1}$, we have $\alpha_{[\mathbf{x}\ 0]} \in (\alpha_{p(\mathbf{x})}, \alpha_{\mathbf{x}})$. Next we consider $E_{[\mathbf{x}\ 1]} = \frac{1}{2}$. As before, the number of 100^α terms decrease, which is negligible. Note the only other change is that an 100^α term is replaced with $(100-o_{i+1})^\alpha$. Therefore, as long as $0 < o_{i+1} < o_i$, then $\alpha_{\mathbf{x}} < \alpha_{[\mathbf{x}\ 1]}$, and similar to the previous case, there exists an upper bound $0 < z'_{i+1} < o_i$ such that for all $0 < o_{i+1} < z'_{i+1}$, we have $\alpha_{[\mathbf{x}\ 1]} \in (\alpha_{\mathbf{x}}, \alpha_{n(\mathbf{x})})$. We conclude that there exists $0 < o_{i+1} < \min(z_i, z'_i) < o_i$ such that $\alpha_{p(\mathbf{x})} < \alpha_{[\mathbf{x}\ 0]} < \alpha_{\mathbf{x}} < \alpha_{[\mathbf{x}\ 1]} < \alpha_{n(\mathbf{x})}$, thus finishing the inductive proof.

Now we have shown that there are $2^{k'}$ nonoverlapping α intervals, such that within every interval, d^α-sampling chooses a unique set of centers, for $k' = k-1$. To finish our structural claim, we will show that after β-Lloyd's method, the cost function $\texttt{clus}_{\alpha,\beta}(\mathcal{V}, \mathbf{Z})$ alternates $2^{k'}$ times above and below a value r as α increases. We add two points, a and b, so that $d(v,a) = d(v,b) = 100$, $d(a_k, a) = d(b_k, b) = 100 - \epsilon$, and the distances from all other points to a and b are length $100 + \epsilon$. Then we add many points in the same location as v, a_i, and b_i, so that any set c returned by d^α-sampling is a local minima for β-Lloyd's method, for all β. Furthermore, these changes do not affect the previous analysis, as long as we appropriately balance the terms in the the numerator and denominator of each equation $E_{\mathbf{x}}$ (and for small enough ϵ). Finally, we set v and a to have label 1 in the target clustering, and all points are labeled 2. Therefore, as d^α-sampling will alternate between $a_k \in C$ and $b_k \notin C$ as we increase α, a and v alternate being in the same or different clusters, so the function $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ will alternate between different outputs $2^{\Omega(n)}$ times as a function of α. □

Now we analyze phase 2 of Algorithm 9. Since phase 2 does not have randomness, we use combinatorial techniques. We define $\texttt{lloyds}_\beta(\mathcal{V}, C, T)$ as the cost of the outputted clustering from phase 2 of Algorithm 9 on instance $\mathcal{V}$ with initial centers C, and a maximum of T iterations.

Theorem 3.4.10. *Given $T \in \mathbb{N}$, a clustering instance $\mathcal{V}$, and a fixed set C of initial centers, the number of discontinuities of $\texttt{lloyds}_\beta(\mathcal{V}, C, T)$ as a function of β is $O(\min(n^{3T}, n^{k+3}))$.*

Proof. Given a clustering instance $\mathcal{V}$ and a vector $\mathbf{Z}$, we bound the number of possible intervals created by the Lloyd's step, given a fixed set of initial centers. Define $\texttt{lloyd}_\beta(\mathcal{V}, C)$ as the cost of the clustering outputted by the β-Lloyd iteration algorithm on $\mathcal{V}$ using initial centers C. Note that the Voronoi partitioning step is only dependent on C, in particular, it is independent of β. Let $\{C_1, \ldots, C_k\}$ denote the Voronoi partition of V induced by C. Given one of these clusters C_i, the next center is computed by $\min_{c \in C_i} \sum_{v \in C_i} d(c, v)^\beta$. Given any $c_1, c_2 \in C_i$, the decision for whether c_1 is a better center than c_2 is governed by $\sum_{v \in C_i} d(c_1, v)^\beta < \sum_{v \in C_i} d(c_2, v)^\beta$. Again by Theorem 3.3.6, this equation has at most $2n+1$ roots. Notice that this equation depends on the set C of centers, the choice of a cluster C_i, and the two points $c_1, c_2 \in C_i$. Then there are $\binom{n}{k} \cdot n \cdot \binom{n}{2}$ total equations which fix the outcome of the Lloyd's method, and there are $\binom{n}{k} \cdot n \cdot \binom{n}{2} \cdot (2n+1) \leq n^{k+4}$ total intervals of β such that the outcome of Lloyd's method is fixed.

Next we give a different analysis which bounds the number of discontinuities by n^{3T}, where T is the maximum number of Lloyd's iterations. By the same analysis as the previous paragraph, if we only consider one round, then the total number of equations which govern the output of a Lloyd's iteration is $\binom{n}{2}$, since the set of centers C is fixed. These equations have $2n + 1$ roots, so the total number of intervals in one round is $O(n^3)$. Therefore, over T rounds, the number of intervals is $O(n^{3T})$. □

By combining Theorem 3.4.8 with Theorem 3.4.10, and using standard learning theory results, we can bound the sample complexity needed to learn near-optimal parameters α, β for an unknown distribution $\mathcal{D}$ over clustering instances. Recall that $\texttt{clus}_{\alpha,\beta}(\mathcal{V})$ denotes the expected cost of the clustering outputted by (α, β)-Lloyds++, with respect to the target clustering, and let H denote the maximum value of $\texttt{clus}_{\alpha,\beta}(\mathcal{V})$.

Theorem 3.4.11. *Given α_h and a sample of size*

$$m = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\min(T, k) \log n + \log \frac{1}{\delta} + \log \alpha_h\right)\right)$$

from $\left(\mathcal{D} \times [0,1]^k\right)^m$, with probability at least $1-\delta$ over the choice of the sample, for all $\alpha \in [0, \alpha_h]$ and $\beta \in [1, \infty) \cup \{\infty\}$, we have

$$\left| \frac{1}{m} \sum_{i=1}^{m} \texttt{clus}_{\alpha,\beta}\left(V^{(i)}, \mathbf{Z}^{(i)}\right) - \mathop{\mathbb{E}}_{V \sim \mathcal{D}} \left[\texttt{clus}_{\alpha,\beta}\left(V\right)\right] \right| < \epsilon.$$

Proof. Fix $\delta > 0$. From Theorems 3.4.8 and 3.4.10, for each i, the expected number of discontinuities of $\texttt{clus}_\alpha(\mathcal{V}_i, Z_i)$ is at most $(8nk \log n) \min(n^k, n^{3T})$. By a Markov inequality, the number of discontinuities is $\leq \frac{1}{\delta} \cdot (16mnk \log n) \min(n^k, n^{3T})$ with probability $\geq 1 - \frac{\delta}{2m}$. Therefore,

Algorithm 10 Dynamic algorithm configuration

Input: Instance $\mathcal{V} = (V, d, k)$, randomness $\mathbf{Z}$, α_h

1. Initialize Q to be an empty queue, then push the root node $(\langle\rangle, [0, \alpha_h])$ onto Q.
2. While Q is non-empty
   (a) Pop node (C, A) from Q with centers C and alpha interval A.
   (b) Compute all points $u_1, \ldots, u_m$ that can be chosen for the next center for some value of $\alpha \in A$.
   (c) For each u_i, set $C_i = C \cup \{u_i\}$ and define $A_i = \{\alpha \in A : u_i \text{ is the sampled center}\}$.
   (d) For each i, if $|C_i| < k$, push (C_i, A_i) onto Q. Otherwise, output (C_i, A_i).

with probability $\geq 1 - \frac{\delta}{2}$, the total number of discontinuities of $\frac{1}{m}\sum_{i=1}^{m} \texttt{clus}_\alpha(\mathcal{V}_i, Z_i)$ over α is $\leq \frac{1}{\delta} \cdot (16m^2nk \log n) \min(n^k, n^{3T})$.

Now we apply Massart's Lemma [Massart, 2000]. Let $N \leq \frac{1}{\delta} \cdot (16m^2nk \log n) \min(n^k, n^{3T})$ denote the number of α-intervals such that $\frac{1}{m}\sum_{i=1}^{m} \texttt{clus}_\alpha(\mathcal{V}_i, Z_i)$ is constant along each interval. For each interval $1 \leq i \leq N$, choose an arbitrary α_i in the interval, and define $a_i = [\texttt{clus}_{\alpha_i}(\mathcal{V}_1, Z_i), \ldots, \texttt{clus}_{\alpha_N}(\mathcal{V}_m, Z_m)]$. Then with probability $\geq 1 - \frac{\delta}{2}$,

$$\hat{R}(A) \leq \max_i ||a_i - \bar{a}|| \cdot \frac{\sqrt{2 \log N}}{m} \leq \sqrt{\frac{2 \log \frac{16m^2nk \log n \min(n^k, n^{3T})}{\delta}}{m}},$$

and the proof follows from standard Rademacher complexity bounds [Bartlett and Mendelson, 2002]. □

Note that a corollary of Theorem 3.4.11 and Lemma 3.4.1 is a uniform convergence bound for *all* $\alpha \in [0, \infty) \cup \{\infty\}$, however, the algorithm designer may decide to set $\alpha_h < \infty$.

3.4.2 Computational efficiency

In this section, we present an algorithm for tuning α whose running time scales with the true number of discontinuities over the sample. Combined with Theorem 3.4.8, this gives a bound on the expected running time of tuning α.

The high-level idea of our algorithm is to directly enumerate the set of centers that can possibly be output by d^α-sampling for a given clustering instance $\mathcal{V}$ and pre-sampled randomness $\mathbf{Z}$. We know from the previous section how to count the number of new breakpoints at any given state in the algorithm, however, efficiently solving for the breakpoints poses a new challenge. From the previous section, we know the breakpoints in α occur when $\frac{D_i(\alpha)}{D_n(\alpha)} = z_t$. This is an exponential equation with n terms, and there is no closed-form solution for α. Although an arbitrary equation of this form may have up to n solutions, our key observation is that if $d_1 \geq \cdots \geq d_n$, then $\frac{D_i(\alpha)}{D_n(\alpha)}$ must be monotone decreasing (from Lemma 3.4.6), therefore, it suffices to binary search over α to find the unique solution to this equation. We cannot find the exact value of the breakpoint from binary search (and even if there was a closed-form solution for the breakpoint, it might not be rational), however we can find the value to within additive error ϵ for all $\epsilon > 0$. Now we show that the expected cost function is $(Hnk \log n)$-Lipschitz in α, therefore, it suffices to run $O\left(\log \frac{Hnk}{\epsilon}\right)$

rounds of binary search to find a solution whose expected cost is within ϵ of the optimal cost. This motivates Algorithm 10. Let $\texttt{seed}_\alpha(\mathcal{V})$ denote the output of d^α sampling run on $\mathcal{V}$.

Lemma 3.4.12. *Given a clustering instance $\mathcal{V}$, $\epsilon > 0$, and $\alpha \in [0, \infty) \cup \{\infty\}$, $P(\texttt{seed}_\alpha(\mathcal{V}) \neq \texttt{seed}_{\alpha+\epsilon}(\mathcal{V})) \leq \epsilon nk \log n$.*

Proof. Given a clustering instance $\mathcal{V}$, $\epsilon > 0$, and a vector $\mathbf{Z} \sim [0, 1]^k$, we will show there is low probability that $\texttt{seed}_\alpha(\mathcal{V}, \mathbf{Z})$ outputs a different set of centers than $\texttt{seed}_{\alpha+\epsilon}(\mathcal{V}, \mathbf{Z})$. Assume in round t of d^α-sampling and $d^{\alpha+\epsilon}$-sampling, both algorithms have C as the current list of centers. Given we draw $z_t \sim [0, 1]$, we will show there is only a small chance that the algorithms choose different centers in this round. Since the algorithms have an identical set of current centers, the distances $d(v, C)$ are the same, but the break points of the intervals, $\frac{\sum_{j=1}^{i} d(v_j,C)^\alpha}{\sum_{j=1}^{n} d(v_j,C)^\alpha}$ differ slightly. If $z_t \sim [0, 1]$ lands in the the same interval I_i and I_i' for d^α-sampling and $d^{\alpha+\epsilon}$-sampling, respectively, then the two algorithms will choose the same point. Thus, we need to bound the size of $\sum_{i=1}^{n}(I_i \setminus I_i') \cup (I_i' \setminus I_i)$. Recall the endpoint of interval i is $\frac{D_i(\alpha)}{D_n(\alpha)}$, where $D_i = \sum_{j=1}^{i} d(v_j, C)^\alpha$. Thus, we want to bound $\left|\frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)}\right|$, and we can use Lemma 3.4.7, which bounds the derivative of $\frac{D_i(\alpha)}{D_n(\alpha)}$ by $(4 \log n)$, to show $\left|\frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)}\right| \leq (4 \log n)\epsilon$.

Therefore, we have

$$\begin{aligned}
&\sum_{i=1}^{n}(I_i \setminus I_i') \cup (I_i' \setminus I_i) && \leq \sum_{i=1}^{n}\left|\frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)}\right| \\
&\leq \sum_{i=1}^{n} \epsilon \cdot (4 \log n) \\
&\leq 4\epsilon n \log n
\end{aligned}$$

Therefore, assuming d^α-sampling and $d^{\alpha+\epsilon}$-sampling have chosen the same centers so far, the probability that they choose different centers in round t is $\leq 4\epsilon n \log n$. Over all rounds, the probability the outputted set of centers is not identical, is $\leq 4\epsilon nk \log n$. □

In order to analyze the runtime of Algorithm 10, we consider the *execution tree* of d^α-sampling run on a clustering instance $\mathcal{V}$ with randomness $\mathbf{Z}$. This is a tree where each node is labeled by a state (i.e., a sequence C of up to k centers chosen so far by the algorithm) and the interval A of α values that would result in the algorithm choosing this sequence of centers. The children of a node correspond to the states that are reachable in a single step (i.e., choosing the next center) for some value of $\alpha \in A$. The tree has depth k, and there is one leaf for each possible sequence of k centers that d^α-sampling will output when run on $\mathcal{V}$ with randomness $\mathbf{Z}$. Our algorithm enumerates these leaves up to an error ϵ in the α values, by performing a depth-first traversal of the tree.

Theorem 3.4.13. *Given a clustering instance $\mathcal{V}$, an interval $[0, \alpha_h]$, $\epsilon > 0$, and $\beta \in [1, \infty) \cup \{\infty\}$, Algorithm 10 outputs a value $\bar{\alpha}$ whose expected cost is within ϵ of the optimal α, i.e., $|\texttt{clus}_{\bar{\alpha},\beta}(\mathcal{V}) - \min_{0 \leq \alpha \leq \alpha_h} \texttt{clus}_{\alpha,\beta}(\mathcal{V})| < \epsilon$. The expected runtime is $O\left(n^2k^2\alpha_h \log\left(\frac{nH}{\epsilon}\right) \log n\right)$.*

Proof. First we establish the correctness of Algorithm 10 (that it outputs $\hat{\alpha}$ such that

$$|\texttt{clus}_{\hat{\alpha},\beta}(\mathcal{V}, \mathbf{Z}) - \texttt{clus}_{\alpha^*,\beta}(\mathcal{V}, \mathbf{Z})| < \epsilon, \text{ where } \alpha^* = \text{argmin}_{0 \le \alpha \le \alpha_h} \texttt{clus}_{\alpha,\beta}(\mathcal{V}, \mathbf{Z}).$$

In each node of the execution tree, Algorithm 10 performs $\log \frac{nkH \log n}{\epsilon}$ rounds of binary search to find each breakpoint to within $\frac{\epsilon}{nkH \log n}$ error. Therefore, the algorithm must output a value $\bar{\alpha}$ such that $|\bar{\alpha} - \hat{\alpha}| < \frac{\epsilon}{nkH \log n}$, where $E[\texttt{clus}_{\hat{\alpha}}(\mathcal{V})] - E[\texttt{clus}_{\alpha^*}(\mathcal{V})] \le \epsilon/2$. By Lemma 3.4.12, the probability that $\hat{\alpha}$ and $\bar{\alpha}$ output different centers is $\le \frac{\epsilon}{2nkH} \cdot nk(\log n)$, therefore, $E[\texttt{clus}_{\bar{\alpha}}(\mathcal{V})] - E[\texttt{clus}_{\hat{\alpha}}(\mathcal{V})] \le \epsilon/2$. Therefore, we conclude that $E[\texttt{clus}_{\bar{\alpha}}(\mathcal{V})] - E[\texttt{clus}_{\alpha^*}] \le \epsilon$.

Now we analyze the runtime of Algorithm 10. Let (C, A) be any node in the algorithm, with centers C and alpha interval $A = [\alpha_\ell, \alpha_h]$. Sorting the points in $\mathcal{V}$ according to their distance to C has complexity $O(n \log n)$. Finding the points sampled by d^α-sampling with α set to α_ℓ and α_h costs $O(n)$ time. Finally, computing the alpha interval A_i for each child node of (C, A) costs $O(n \log \frac{nH}{\epsilon})$ time, since we need to perform $\log \frac{nkH \log n}{\epsilon}$ iterations of binary search on $\alpha \mapsto \frac{D_i(\alpha)}{D_n(\alpha)}$ and each evaluation of the function costs $O(n)$ time. We charge this $O(n \log \frac{nH}{\epsilon})$ time to the corresponding child node. If there are N nodes in the execution tree, summing this cost over all nodes gives a total running time of $O(N \cdot n \log \frac{nH}{\epsilon}))$. If we let $\#I$ denote the total number of α-intervals for $\mathcal{V}$, then each layer of the execution tree has at most $\#I$ nodes, and the depth is k, giving a total running time of $O(\#I \cdot kn \log \frac{nH}{\epsilon})$.

From Theorem 3.4.8, we have $\mathbb{E}[\#I] \le 8nk \log n \cdot \alpha_h$. Therefore, the expected runtime of Algorithm 10 is $O\left(n^2 k^2 \alpha_h (\log n)\left(\log \frac{nH}{\epsilon}\right)\right)$. This completes the proof. □

Since we showed that d^α-sampling is Lipschitz as a function of α in Lemma 3.4.12, it is also possible to find the best α parameter with sub-optimality at most ϵ by finding the best point from a discretization of $[0, \alpha_h]$ with step-size $s = \epsilon/(Hn^2 k \log n)$. The running time of this algorithm is $O(n^3 k^2 H \log n/\epsilon)$, which is significantly slower than the efficient algorithm presented in this section. Intuitively, Algorithm 10 is able to binary search to find each breakpoint in time $O(\log \frac{nH}{\epsilon})$, whereas a discretization-based algorithm must check all values of alpha uniformly, so the runtime of the discretization-based algorithm increases by a multiplicative factor of $O\left(\frac{nH}{\epsilon} \cdot \left(\log \frac{nH}{\epsilon}\right)^{-1}\right)$.

Chapter 4

Data-Driven Dispatching for Distributed 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 [Balcan et al., 2012, 2013b, Liang et al., 2014, Li et al., 2014, Zhang et al., 2013, 2012]. 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 [Li et al., 2014, Zhang et al., 2012]. 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 [Zhang et al., 2012, 2013]. 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" [Vapnik and Bottou, 1993], 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

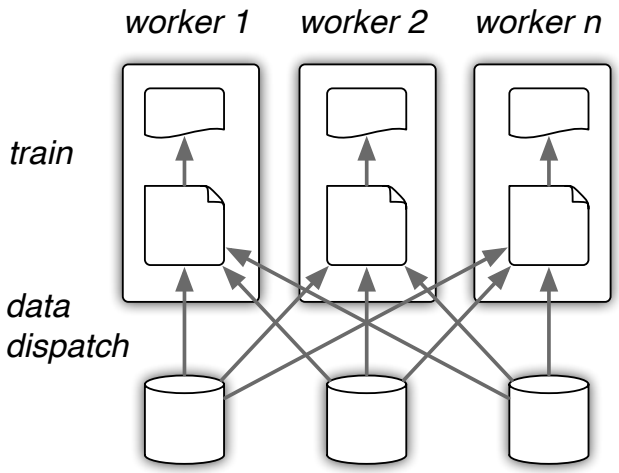


Figure 4.1: Data is partitioned and dispatched to multiple workers. Each worker then trains a local model using its local data. There is no communication between workers during training.

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 [Balcan et al., 2013c, Bateni et al., 2014b, Malkomes et al., 2015]. 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 [Balcan et al., 2013c, Bateni et al., 2014b, Malkomes et al., 2015]. For example, Malkomes et al. provide distributed algorithms for k-center and k-center with outliers Malkomes et al. [2015], and Bateni et al. introduce distributed algorithms for capacitated k-clustering under any ℓ_p objective Bateni et al. [2014b]. 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 this chapter, 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.

4.1.1 Results and techniques

We propose a novel scheme for partitioning data which leads to better accuracy in distributed machine learning tasks, and we give a theoretical analysis of this approach. We present new algorithms with provable worst-case guarantees for dispatching in realistic settings. Empiricallyk, our method strongly scales and that we achieve significantly higher accuracy over baselines based on random partitioning, balanced partition trees, and locality-sensitive hashing Dick et al. [2017].

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. Finally, each machine independently learns a classifier using its own data (in an embarrassingly parallel manner). To perform the initial clustering used for dispatch, we use classic clustering objectives (k-means, k-median, and k-center). However, we need to add novel constraints to ensure that the clusters give a data partition that respects the constraints of real distributed learning systems:

Balancedness: We need to ensure our dispatching procedure balances the data across the different machines. If a machine receives much more data than other machines, then it will be the bottleneck of the algorithm. If any machine receives very little data, then its processing power is wasted. Thus, enforcing upper and lower bound constraints on the cluster sizes leads to a faster, more efficient setup.

Fault-Tolerance: In order to ensure that our system is robust to machine failures, we assign each point to multiple distinct clusters. This way, even if a machine fails, the data on that machine is still present on other machines. Moreover, this has the added benefit that our algorithms behave well on points near the boundaries of the clusters. We say a clustering algorithm satisfies p-replication if each point is assigned to p distinct clusters.

Efficiency: To improve efficiency, we apply our clustering algorithms to a small sample of data. Therefore, we need to be able to extend the clustering to new examples from the same distribution while maintaining a good objective value and satisfying all constraints. The extension technique should be efficient for both the initial partitioning, and dispatching at prediction time.

When designing clustering algorithms, adding balancedness and fault tolerance makes the task significantly harder. Prior work has considered upper bounds on the cluster sizes [An et al., 2014, Byrka et al., 2015b, Cygan et al., 2012, Khuller and Sussmann, 1996, Li, 2014, 2016] and lower bounds [Aggarwal et al., 2006, Ahmadian and Swamy, 2016], but no prior work has shown provable guarantees with upper and lower bounds on the cluster sizes simultaneously. [1] With upper bounds, the objective functions are nondecreasing as the number of clusters k increases, but with lower bounds we show the objective function can oscillate arbitrarily with respect to k. This makes the problem especially challenging from a combinatorial optimization perspective. Existing capacitated clustering algorithms work by rounding a fractional linear program solution, but the erratic nature of the objective function makes this task more difficult for us.

[1] Note that enforcing only upper (resp. lower) bounds implies a weak lower (resp. upper) bound on the cluster sizes, but this is only nontrivial if the upper (resp. lower) bounds are extremely tight or the number of clusters is a small constant.

The balance constraints also introduce challenges when extending a clustering-based partitioning from a small sample to unseen data. The simple rule that assigns a new point to the cluster with the nearest center provides the best objective value on new data, but it can severely violate the balance constraints. Therefore, any balanced extension rule must take into account the distribution of data.

We overcome these challenges, presenting a variety of complementary results, which together provide strong justification for our distributed learning framework. We summarize each of our main results below.

Balanced fault-tolerant clustering. We provide the first clustering algorithms with provable guarantees that simultaneously handle upper and lower bounds on the cluster sizes, as well as fault tolerance. Clustering is NP-hard and adding more constraints makes it significantly harder, as we will see in Section 4.4. For this reason, we first devise approximation algorithms with strong worst-case guarantees, demonstrating this problem is tractable. Specifically, in Section 4.3 we provide an algorithm that produces a fault-tolerant clustering that approximately optimizes k-means, k-median, and k-center objectives while also roughly satisfying the given upper and lower bound constraints. At a high level, our algorithm proceeds by first solving a linear program, followed by a careful balance and replication aware rounding scheme. We use a novel min-cost flow technique to finish off rounding the LP solution into a valid clustering solution. No previous work gives provable guarantees while satisfying both upper and lower bounds on the cluster sizes, and Section 4.3 may be of independent interest beyond distributed learning.

Structure of balanced clustering. We show that adding lower bound constraints on the cluster sizes makes clustering objectives highly nontrivial. Specifically, we show that for k-means, k-median, and k-center, the objective values may oscillate arbitrarily with respect to k. In light of this structure, our aforementioned algorithmic results are more surprising, since it is not obvious that algorithms with constant-factor guarantees exist.

General Robust Distributed Clustering. In Section 4.5, we show a general distributed algorithm for balanced k-clustering in ℓ_p in d dimensions with z outliers, using $O(m(k+z)(d+\log n))$ communication. The algorithm can be summarized as follows. Each machine performs a k-clustering on its own data, and then sends the centers, along with the sizes of their corresponding clusters, to a central machine. The central machine then runs a weighted clustering algorithm on the mk centers. We add necessary changes to handle capacities and outliers, for instance, each (non-central) machine runs a $k+z$ clustering algorithm in the case of k-clustering with z outliers. Given a sequential α-approximation algorithm and a bicriteria β-approximation algorithm which opens up $\tilde{O}(k)$ centers, we show our distributed algorithm returns an $O(\alpha\beta)$ approximation and improves over the approximation guarantees of Bateni et al. [2014b]. For example, for k-median, we achieve a $(6\alpha+2+\epsilon)$-approximation by plugging in the bicriteria algorithm of Lin and Vitter [1992] (adding an $O(\log n)$ factor to the communication cost), as opposed to a 32α-approximation from Bateni et al. [2014b]. By plugging in the approximation algorithm for k-median with outliers [Chen, 2008], our algorithm achieves the first constant approximation for distributed k-median with outliers, answering an open question posed by Malkomes et al. [2015]. We achieve these improvements by using a refined analysis to prove a strong bound on the distance between each local center and its closest center in the solution outputted by the algorithm. We show how to carefully

reason about the optimal clustering with subsets of outliers removed to preserve the constant factor approximation guarantee for k-median with outliers.

4.1.2 Related work

Distributed Learning. Currently, the most popular method of dispatch in distributed learning is random dispatch [Zhang et al., 2012, 2013]. This may not produce optimal results because each machine must learn a global model. Another notion is to dispatch the data to pre-determined locations e.g., Yahoo!'s geographically distributed database, PNUTS [Cooper et al., 2008]. However, it does not look at any properties of the data other than physical location.

In a recent paper, Wei et al. [2015] study partitioning for distributed machine learning, however, they give no formal guarantees on balancing the data each machine receives. You et al. [2015] use k-means clustering to distribute data for parallel training of support vector machines, but their clustering algorithms do not have approximation guarantees and are applied to the entire dataset, so their clustering step is much slower than ours. There is also work on distributed graph partitioning [Aydin et al., 2016, Bourse et al., 2014, Delling et al., 2011], in which the data points are set up in a graph structure, and must be distributed to different machines, minimizing the number of edges across machines. These techniques do not apply more generally for non graph-based objectives, e.g. k-means, k-median, or k-center.

Centralized Clustering. The first constant-factor approximation algorithm for k-median was given by Charikar et al. [1999b], and the current best approximation ratio is 2.675 from Byrka et al. [2015c]. For k-center, there is a tight 2-approximation algorithm [Gonzalez, 1985]. For k-means, the best approximation ratio is 6.357 [Ahmadian et al., 2017], and Makarychev et al. [2016] recently showed a bicriteria algorithm with strong guarantees. For clustering with outliers, there is a 3-approximation algorithm for k-center with z outliers, as well as a bicriteria $4(1 + 1/\epsilon)$-approximation algorithm for k-median that picks $(1 + \epsilon)z$ outliers Charikar et al. [2001]. Chen found a true constant factor approximation algorithm for k-median (the constant is not explicitly computed) [Chen, 2008].

Capacitated k-center The (uniform) capacitated k-center problem is to minimize the maximum distance between a cluster center and any point in its cluster subject to the constraint that the maximum size of a cluster is L. It is NP-Hard, so research has focused on finding approximation algorithms. Bar-Ilan et al. [1993] introduced the problem and presented the first constant factor polynomial time algorithm achieving a factor of 10, using a combinatorial algorithm which moves around clients until the capacities are satisfied, and the objective is approximately satisfied. The approximation factor was improved by Khuller and Sussmann [1996]. Cygan et al. [2012] give the first algorithm for capacitated k-center with non-uniform capacities by using an LP rounding algorithm. The appoximation factor is not explicitly computed, although it is mentioned to be in the order of hundreds. An et al. [2014] follow a similar procedure but with a dynamic rounding procedure, and they improve to an approximation factor of 8. Further, for the special case of uniform capacities, they show a 6-approximation.

Capacitated k-median k-median with capacities is a notoriously difficult problem in clustering. It is much less understood than k-center with capacities, and uncapacitated k-median, both of which have constant factor approximations. Despite numerous attempts by various researchers,

still there is no known constant factor approximation for capacitated k-median (even though there is no better lower bound for the problem than the one for uncapacitated k-median). As stated earlier, there is a well-known unbounded integrality gap for the standard LP even when violating the capacity or center constraints by a factor of $2 - \epsilon$ [Aardal et al., 2015].

Charikar et al. gave a 16-approximation when constraints are violated by a factor of 3 [Charikar et al., 1999a]. Byrka et al. improved this violation to $2+\epsilon$, while maintaining an $O(\frac{1}{\epsilon^2})$ approximation [Byrka et al., 2015a]. Recently, Li improved the latter to $O(\frac{1}{\epsilon})$, specifically, when constraints are violated by $2 + \frac{2}{\alpha}$ for $\alpha \geq 4$, they give a $6 + 10\alpha$ approximation [Li, 2014]. These results are all for the *hard* capacitated k-median problem. In the *soft* capacities variant, we can open a point more than once to achieve more capacity, although each extra opening counts toward the budget of k centers. In hard capacities, each center can only be opened once. The hard capacitated version is more general, as each center can be replicated enough times so that the soft capacitated case reduces to the hard capacitated case. Therefore, we will only discuss the hard capacitated case.

All of the algorithms for capacitated k-median mentioned above share the same high-level LP rounding and aggregation idea but with different refinements in the algorithm and analysis.

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 Balcan et al. [2013c]. Bateni et al. indroduced a construction for *mapping coresets*, which admits a distributed clustering algorithm that can handle balance constraints with communication cost $\tilde{O}(mk)$ [Bateni et al., 2014b]. Malkomes et al. showed a distributed 13- and 4- approximation algorithm for k-center with and without outliers, respectively [Malkomes et al., 2015]. 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 [Chen et al., 2016], building on a recent lower bound for set-disjointness in the message-passing model [Braverman et al., 2013]. Garg et al. showed a communication complexity lower bound for computing the mean of d-dimensional points [Garg et al., 2014].

4.2 Preliminaries

Clustering. Recall the formal definition of clustering stated in previous chapters. Given a set of points V of size n and a distance metric d, let $\mathcal{C}$ denote a clustering of V, which we define as a partition of V into k subsets $X_1, \ldots, X_k$. Each cluster X_i contains a center x_i. When d is an arbitrary distance metric, we must choose the centers from the point set. If $V \subseteq \mathbb{R}^d$ and the distance metric is the standard Euclidean distance, then the centers can be any k points in $\mathbb{R}^d$. In fact, this distinction only changes the cost of the optimal clustering by at most a factor of 2 when $p = 1, 2$, or ∞ [Awasthi and Balcan, 2014]. The ℓ_p cost of $\mathcal{C}$ is

$$\text{cost}(\mathcal{C}) = \left(\sum_i \sum_{v \in X_i} d(v, x_i)^p \right)^{\frac{1}{p}}.$$

We will denote the optimal clustering of a point set V in ℓ_p with z outliers as $\mathcal{OPT}_{k,z,p}(V)$. V, p, k, and/or z will often be clear from context, so we may drop some or all of these parameters. $\mathcal{OPT}(A, B)$ will denote the optimal clustering for a point set $A \subseteq V$, using centers from a different

point set $B \subseteq V$. We often overload notation and let $\mathcal{OPT}$ denote the objective value of the optimal clustering as well. In our proofs, we make use of the triangle inequality generalized for ℓ_p, $d(u,v)^p \leq 2^{p-1}(d(u,w)^p + d(w,v)^p)$, for any points u, v, w. We denote the optimal clusters as $C_1, \ldots, C_k$, with centers $c_1, \ldots, c_k$. We say a bicriteria clustering algorithm $\mathcal{A}$ is a (γ, α)-approximation algorithm if it returns $\gamma \cdot k$ centers which define a clustering whose cost is at most an α-factor from the optimal clustering with k centers. Throughout the section, unless otherwise noted, we assume any d-dimensional datapoint can be expressed using $O(d)$ bits.

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 [Braverman et al., 2013]. 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 [Bateni et al., 2014b, Malkomes et al., 2015].

4.3 Fault Tolerant Balanced Clustering

In this section, we give an algorithm to cluster a small initial sample of data to create a dispatch rule that sends similar points to the same machine. There are many ways to measure the similarity of points in the same cluster. We consider three classic clustering objectives while imposing upper and lower bounds on the cluster sizes and replication constraints. It is well-known that solving the objectives optimally are NP-hard even without the capacity and fault tolerance generalizations [Jain et al., 2003]. In Section 4.4, we show that the objectives with balance constraints behave erratically with respect to the number of clusters k, in particular, there may exist an arbitrary number of local minima and maxima. In light of this difficulty, one might ask whether any approximation algorithm is possible for this problem. We answer affirmatively, by extending previous work [Li, 2014] 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 and replication constraints by a small constant factor. This is the first algorithm with provable guarantees to simultaneously handle both upper and lower bounds on the cluster sizes.

Theorem 4.3.1. *Algorithm 4.1 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.*

Recall that formally, a clustering instance consists of a set V of n points, and a distance metric d. Given two points i and j in V, denote the distance between i and j by $d(i,j)$. The task is to find a set of k centers $C = \{c_1, \ldots, c_k\}$ and assignments of each point to p of the centers $f : V \to \binom{C}{p}$, where $\binom{C}{p}$ represents the subset of C^p with no duplicates. In this section, we study k-median, k-means, and k-center. Due to the new p-replication constriant, we formally state these objectives below.

(1) k-median: $\min_{C,f} \sum_{i \in V} \sum_{j \in f(i)} d(i,j)$

(2) k-means: $\min_{C,f} \sum_{i \in V} \sum_{j \in f(i)} d(i,j)^2$

(3) k-center: $\min_{C,f} \max_{i \in V} \max_{j \in f(i)} d(i,j)$

We add size constraints $0 < \ell \leq L < 1$, also known as capacity constraints, so each cluster must have a size between $n\ell$ and nL. For simplicity, we assume these values are integral (or replace them by $\lceil n\ell \rceil$ and $\lfloor nL \rfloor$ respectively). First we consider k-median and k-means, and later we turn to k-center.

4.3.1 Bicriteria algorithms

At a high level, our algorithm proceeds by first solving a linear program, followed by careful rounding. In particular, we set up an LP whose optimal integral solution is the optimal clustering. We can use an LP solver which will give a fractional solution (for example, the LP may open up $2k$ half centers). Then, using a greedy procedure from Charikar et al. [1999a], we pick $\leq k$ points (called the 'monarchs) which are spread out. Furthermore, the distance from a non-monarch to its closest monarch is a constant-factor multiple of the non-monarch's connection cost in the LP solution. The empire of a monarch is defined to be its cell in the Voronoi partition induced by the monarchs. By a Markov inequality, every empire has $\geq p/2$ total fractional centers, which is at least one for $p \geq 2$. Then we merely open the innermost points in the empires as centers, ending with $\leq k$ centers. Once we have the centers, we find the optimal assignments by setting up a min-cost flow problem.

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 procedure is summarized in Algorithm 4.1.

Step 1: Linear Program The first step is to solve a linear program (LP) relaxation of the standard integer program (IP) formulation of our constrained clustering problem. 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. In the LP, the variables may be fractional, so y_i represents the fraction to which a center is opened (we will refer to this as the "opening" of i), and x_{ij} represents the fractional assignment of j to i. One can use an LP solver to get a fractional solution which must then be rounded. Let (x, y) denote an optimal solution to the LP. For any points i and j, let c_{ij} be the cost of assigning point j to center i. That is, for k-median, $c_{ij} = d(i,j)$, and for k-means $c_{ij} = d(i,j)^2$ (we discuss k-center in the next section). Define $C_j = \sum_i c_{ij} x_{ij}$, the average cost from point j to its centers in the LP solution (x, y).

It is well-known that the LP in Algorithm 4.1 has an unbounded integrality gap (the ratio of the optimal LP solution over the optimal integral LP solution), even when the capacities are violated by a factor of $2 - \epsilon$ [Li, 2014]. We give the integrality gap below. 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.

Step 1 details We restate the LP for k-means and k-median for completeness, labeling each constraint for the proofs later.

1. Find a solution to the following linear program:

$$\min_{x,y} \sum_{i,j \in V} c_{ij} x_{ij} \quad \text{s.t.} \quad \textbf{(a)} \, \forall j \in V : \sum_{i \in V} x_{ij} = p; \qquad \textbf{(b)} \sum_{i \in V} y_i \leq k;$$

$$\textbf{(c)} \, \forall i \in V : \ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i; \quad \textbf{(d)} \, \forall i, j \in V : 0 \leq x_{ij} \leq y_i \leq 1.$$

2. Greedily place points into a set $\mathcal{M}$ from lowest C_j to highest (called "monarchs"), adding point j to $\mathcal{M}$ if it is not within distance $4C_j$ of any monarch. Partiton the points into coarse clusters (called "empires") using the Voronoi partitioning of the monarchs.
3. For each empire $\mathcal{E}_u$ with total fractional opening $Y_u \triangleq \sum_{i \in \mathcal{E}_u} y_i$, give opening $Y_u / \lfloor Y_u \rfloor$ to the $\lfloor Y_u \rfloor$ closest points to u and all other points opening 0.
4. Round the x_{ij}'s by constructing a minimum cost flow problem on a bipartite graph of centers and points, setting up demands and capacities to handle the bounds on cluster sizes.

Algorithm 4.1: Balanced clustering with fault tolerance

$$\min \sum_{i,j \in V} c_{ij} x_{ij} \tag{LP.1}$$

$$\text{subject to:} \sum_{i \in V} x_{ij} = p, \ \forall j \in V \tag{LP.2}$$

$$\ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i, \ \forall i \in V \tag{LP.3}$$

$$\sum_{i \in V} y_i \leq k; \tag{LP.4}$$

$$0 \leq x_{ij} \leq y_i \leq 1, \ \forall i, j \in V. \tag{LP.5}$$

It is well-known that the standard capacitated k-median LP (this LP, without the lower bound constraint and with $p = 1$) has an unbounded integrality gap, even when the capacities are violated by a factor of $2 - \epsilon$ [Aardal et al., 2015]. The integrality gap is as follows. $k = 2nL - 1$, and there are nL groups of size $2nL - 1$. Points in the same group are distance 0, and points in different groups are distance 1. Fractionally, we can open $2 - \frac{1}{nL}$ facilities in each group to achieve cost 0. But integrally, some group contains at most 1 facility, and thus the capacity violation must be $2 - \frac{1}{nL}$.

However, with p replication, there must be p centers per group, so the balance violation can be split among the p centers. Therefore, the integrality is only unbounded when the capacities are violated by a factor of $\frac{p}{p-1}$

The k-center LP is a little different from the k-median/means LP. As in prior work [An et al., 2014, Cygan et al., 2012, Khuller and Sussmann, 1996], we guess the optimal radius, t. Since there are a polynomial number of choices for t, we can try all of them to find the minimum possible t for which the following program is feasible. Here is the LP for k-center.

Table 4.1: Notation table

Symbol	Description	k-median	k-means	k-center
y_i	Fractional opening at center i	-		
x_{ij}	Fractional assignment of point j to center i	-		
c_{ij}	Cost of assigning j to center i	$d(i,j)$	$d(i,j)^2$	t
C_j	Avg cost of assignment of point j to all its centers	$\sum_i c_{ij}x_{ij}/p$		t
C_{LP}	Cost of LP	$\sum_j pC_j$		
ρ	parameter for monarch procedure	2	4	1

$$\sum_{i\in V} x_{ij} = p, \qquad \forall j \in V \tag{4.2a}$$

$$n\ell y_i \le \sum_{j\in V} x_{ij} \le nLy_i, \qquad \forall i \in V \tag{4.2b}$$

$$\sum_{i\in V} y_i \le k; \tag{4.2c}$$

$$0 \le x_{ij} \le y_i, \qquad \forall i,j \in V \tag{4.2d}$$

$$x_{ij} = 0 \qquad \text{if } d(i,j) > t. \tag{4.2e}$$

For k-median and k-means, let C_{LP} denote the objective value. For k-center, C_{LP} would be the smallest threshold t at which the LP is feasible, however we scale it as $C_{LP} = tnp$ for consistency with the other objectives. For all $j \in V$, define the connection cost C_j as the average contribution of a point to the objective. For k-median and k-means, it is $C_j = \frac{1}{p}\sum_{i\in V} c_{ij}x_{ij}$. That is, for k-median, it is the average distance of a point to its fractional centers while for k-means, it is the average squared distance of a point to its fractional centers. For k-center, C_j is simply the threshold $C_j = t$. Therefore, $C_{LP} = \sum_{j\in V} pC_j$ in all cases.

The notation is summarized in table 4.1.

Step 2: Monarch Procedure Next, partition the points into "empires" such that every point is $\le 4C_j$ from the center of its empire (the "monarch") by using a greedy procedure from Charikar et al. [1999a] (for an informal description, see step 2 of Algorithm 4.1). By Markov's inequality, every empire has total opening $\ge p/2$, which is crucially ≥ 1 for $p \ge 2$ under our model of fault tolerance. We obtain the following guarantees.

Lemma 4.3.2. *The output of the monarch procedure satisfies the following properties:*

(1a) The clusters partition the point set;

(1b) Each point is close to its monarch: $\forall j \in \mathcal{E}_u, u \in \mathcal{M}, c_{uj} \le 4C_j$*;*

(1c) Any two monarchs are far apart: $\forall u, u' \in \mathcal{M}$ *s.t.* $u \neq u', c_{uu'} > 4\max\{C_u, C_{u'}\}$*;*

(1d) Each empire has a minimum total opening: $\forall u \in \mathcal{M}, \sum_{j \in \mathcal{E}_u} y_j \geq \frac{p}{2}$.

Proof sketch. The first three properties follow easily from construction (for property (1c), recall we greedily picked monarchs by the value of C_j). For the final property, note that for some $u \in \mathcal{M}$, if $d(i,u) \leq 2C_u$, then $i \in \mathcal{E}_u$ (from the triangle inequality and property (1c)). Now, note that C_u is a weighted average of costs c_{iu} with weights x_{iu}/p, i.e., $C_u = \sum_i c_{iu} x_{iu}/p$. By Markov's inequality, in any weighted average, values greater than twice the average have to get less than half the total weight. That is,

$$\sum_{j:c_{ju}>2C_u} \frac{x_{ju}}{p} < \sum_{j:c_{ju}>2C_u} \frac{x_{ju}}{p} \cdot \frac{c_{ju}}{2C_u} < \frac{C_u}{2C_u} = \frac{1}{2}$$

Combining these two facts, for each $u \in \mathcal{M}$:

$$\sum_{j \in \mathcal{E}_u} y_j \geq \sum_{j:c_{ju} \leq 2C_u} y_j \geq \sum_{j:c_{ju} \leq 2C_u} x_{ju} \geq \frac{p}{2}. \quad \square$$

Step 2 details Let $\mathcal{M}$ be the set of monarchs, and for each $u \in \mathcal{M}$, denote $\mathcal{E}_u$ as the empire of monarch u. Recall that the contribution of an assignment to the objective c_{ij} is $d(i,j)$ for k-median, $d(i,j)^2$ for k-means, and t for k-center. We also define a parameter $\rho = 1$ for k-center, $\rho = 2$ for k-median, and $\rho = 4$ for k-means, for convenience.

Initially set $\mathcal{M} = \emptyset$. Order all points in nondecreasing order of C_i. For each point i, if $\exists j \in \mathcal{M}$ such that $c_{ij} \leq 2tC_i$, continue. Else, set $\mathcal{M} = \mathcal{M} \cup \{i\}$. At the end of the for loop, assign each point i to cluster $\mathcal{E}_u$ such that u is the closest point in $\mathcal{M}$ to i. See Algorithm 11.

Algorithm 11 Monarch procedure for coarse clustering: Greedy algorithm to create monarchs and assign empires

Input: V and fractional (x, y)
$\mathcal{M} \leftarrow \emptyset$
Order all points in non-decreasing order of C_i
Identify Monarchs
For each $i \in V$

- **If** $\nexists j \in \mathcal{M}$ such that $c_{ij} \leq 2\rho C_i$
  - $\mathcal{M} \leftarrow \mathcal{M} \cup \{i\}$

Assign Empires as Voronoi partitions around monarchs
For each $j \in V$

- Let $u \in \mathcal{M}$ be the closest monarch to j
- $\mathcal{E}_u \leftarrow \mathcal{E}_u \cup \{j\}$

Output: Set of monarchs, $\mathcal{M}$, and empire $\mathcal{E}_j$ for each monarch $j \in \mathcal{M}$

Now we give the full proof of Lemma 4.3.2.

Proof of Lemma 4.3.2. The first three properties follow easily from construction (for the third property, recall we ordered the points at the start of the monarch procedure). Here is the proof of the final property, depending on the objective function.

For k-center and k-median, it is clear that for some $u \in \mathcal{M}$, if $d(i,u) \leq \rho C_u$, then $i \in \mathcal{E}_u$ (from the triangle inequality and Property (1c)). For k-means, however: if $d(i,u)^2 \leq 2C_u$, then $i \in \mathcal{E}_u$. Note that the factor is $\rho/2$ for k-means. This is because of the triangle inequality is a little different for squared distances.

To see why this is true for k-means, assume towards contradiction that $\exists i \in V$, $u, u' \in \mathcal{M}$, $u \neq u'$ such that $u \in \mathcal{E}_{u'}$ and $d(i,u)^2 \leq 2C_u$. Then $d(i,u') \leq d(i,u)$ by construction. Therefore, $d(u,u')^2 \leq (d(u,i) + d(i,u'))^2 \leq 4d(i,u)^2 \leq 8C_u$, and we have reached a contradiction by Property (1c).

Now, to prove property (1d):

k-center From the LP constraints, for every u, $\sum_{j \in V} x_{ju} = p$. But x_{ju} is non-zero only they are separated by at most t, the threshold. Combining this with the fact that if $d(j,u) \leq C_u = t$, then $j \in \mathcal{E}_u$, we get, for each $u \in \mathcal{M}$:

$$\sum_{j \in \mathcal{E}_u} y_j \geq \sum_{j \in \mathcal{E}_u} x_{ju} = p$$

k-median and k-means Note that C_u is a weighted average of costs c_{iu} with weights x_{iu}/p, i.e., $C_u = \sum_i c_{iu} {}^{x_{iu}}/_p$. By Markov's inequality,

$$\sum_{j : c_{ju} > 2C_u} \frac{x_{ju}}{p} < \frac{C_u}{2C_u} = \frac{1}{2}$$

Combining this with the fact that if $c_{ju} \leq 2C_u$, then $j \in \mathcal{E}_u$ for both k-median and k-means , we get, for each $u \in \mathcal{M}$:

$$\sum_{j \in \mathcal{E}_u} y_j \geq \sum_{j : c_{ju} \leq 2C_u} y_j \geq \sum_{j : c_{ju} \leq 2C_u} x_{ju} \geq \frac{p}{2}.$$

□

Step 3: Aggregation The point of this step is to end up with $\leq k$ centers total. Since each empire has total opening at least 1, we can aggregate openings within each empire. For each empire $\mathcal{E}_u$, we move the openings to the $\lfloor Y_u \rfloor$ innermost points of $\mathcal{E}_u$, where $Y_u = \sum_{i \in \mathcal{E}_u} y_i$. This shuffling is accomplished by greedily calling a suboperation called *Move*, which is the standard way to transfer openings between points to maintain all LP constraints [Li, 2014]. To perform a *Move* from i to j of size δ, set $y'_i = y_i - \delta$ and $y'_j = y_j + \delta$, and change all x's so that the fractional demand switches from i to j: $\forall u \in V$, $x'_{iu} = x_{iu}(1 - {}^{\delta}/_{y_i})$ and similarly increase the demand to all x_{ju}. The *Move* operation preserves all LP constraints, except we may violate the capacity constraints if we give a center an opening greater than one.

In each empire $\mathcal{E}_u$, start with the point i with nonzero y_i that is farthest away from the monarch u. Move its opening to the monarch u, and then iteratively continue with the next-farthest point in $\mathcal{E}_u$ with nonzero opening. Continue this process until u has opening exactly $\frac{Y_u}{\lfloor Y_u \rfloor}$, and then start

moving the farthest openings to the point j closest to the monarch u. Continue this until the $\lfloor Y_u \rfloor$ closest points to u all have opening $\frac{Y_u}{\lfloor Y_u \rfloor}$. Call the new variables (x', y'). They have the following properties.

Lemma 4.3.3. *The aggregated solution (x', y') satisfies the following constraints:*

(2a) The opening of each point is either zero or in $[1, \frac{p+2}{2}]$: $\forall i \in V,\ 1 \le y'_i < \frac{p+2}{p}$ or $y'_i = 0$;

(2b) Each cluster satisfies the capacity constraints: $i \in V,\ \ell y'_i \le \sum_{j \in V} \frac{x'_{ij}}{n} \le L y'_i$;

(2c) The total fractional opening is k: $\sum_{i \in V} y'_i = k$;

(2d) Points are only assigned to open centers: $\forall i, j \in V,\ x'_{ij} \le y'_i$;

(2e) Each point is assigned to p centers: $\forall i \in V,\ \sum_j x'_{ji} = p$;

(2f) The number of points with non-zero opening is at most k: $|\{i \mid y'_i > 0\}| \le k$.

Proof. For the first property, recall that each cluster $\mathcal{E}_u$ has total opening $\ge \frac{p}{2}$, so by construction, all i with nonzero y'_i has $y'_i \ge 1$. We also have $\frac{Y_u}{\lfloor Y_u \rfloor} \le \frac{\lfloor Y_u \rfloor + 1}{\lfloor Y_u \rfloor} \le \frac{p+2}{p}$, which gives the desired bound.

The next four properties are checking that the LP constraints are still satisfied (except for $y'_i \le 1$). These follow from the fact that *Move* does not violate the constraints. The last property is a direct result of Properties (2a) and (2c). □

Now we state the following guarantee about the moving costs.

Lemma 4.3.4. *$\forall j \in V$ whose opening moved from i' to i,*

- *k-median: $d(i, j) \le 3d(i', j) + 8C_j$,*
- *k-means: $d(i, j)^2 \le 15d(i', j)^2 + 80C_j$.*

Below, we formally define the "Move" operation, which allows us to prove Lemma 4.3.4.

Step 3 Details First we define the suboperation Move [Li, 2014]:

Definition 4.3.5 (Operation "Move")**.** *The operation "Move" moves a certain opening δ from a to b. Let (x', y') be the updated (x, y) after a movement of $\delta \le y_a$ from a to b. Define*

$$\begin{aligned} y'_a &= y_a - \delta \\ y'_b &= y_b + \delta \\ \forall u \in V,\ x'_{au} &= x_{au}(1 - \delta/y_a) \\ \forall u \in V,\ x'_{bu} &= x_{bu} + x_{au} \cdot \delta/y_a \end{aligned}$$

It has been proven in previous work that the move operation does not violate any of the LP constraints except the constraint that $y_i \le 1$ [Li, 2014]. We provide a proof below for completeness. Should we require $\delta \le \min(y_a, 1 - y_b)$, the constraint $y_i \le 1$ would not be violated. But to get a bicriteria approximation, we allow this violation. The amount by which the objective gets worse can then be bounded by the triangle inequality.

Lemma 4.3.6. *The operation Move does not violate any of the LP constraints except possibly the constraint $y_i \leq 1$ and the threshold constraint 4.2e of k-center.*

Proof. To show that the *Move* operation satisfies all the LP constraints, first note that the only quantities that change are $y_a, y_b, x_{au}, x_{bu},\ \forall u \in V$. Further, x, y satisfy all the constraints of the LP. Using this,

- Constraint LP.1: For every u, $\sum_i x'_{iu} = \sum_i x_{iu} = p$.
- Constraint LP.2 (1):

$$\begin{aligned}\sum_u x'_{au} &= \sum_u x_{au}(1 - \delta/y_a) \leq nLy_a(1 - \delta/y_a) = nLy'_a \\ \sum_u x'_{bu} &= \sum_u x_{bu} + \sum_u x_{au} \cdot \delta/y_a \\ &\leq nLy_b + nLy_a \cdot \delta/y_a = nLy'_b\end{aligned}$$

- Constraint LP.2 (2):

$$\begin{aligned}\sum_u x'_{au} &= \sum_u x_{au}(1 - \delta/y_a) \geq n\ell y_a(1 - \delta/y_a) = n\ell y'_a \\ \sum_u x'_{bu} &= \sum_u x_{bu} + \sum_u x_{au} \cdot \delta/y_a \\ &\geq n\ell y_b + n\ell y_a \cdot \delta/y_a = n\ell y'_b\end{aligned}$$

- Constraint LP.3: $\sum_i y'_i = \sum_i y_i \leq k$
- Constraint LP.4 (1):

$$\begin{aligned}x'_{au} &= x_{au}(1 - \delta/y_a) \leq y_a(1 - \delta/y_a) = y'_a \\ x'_{bu} &= x_{bu} + x_{au} \cdot \delta/y_a \leq y_b + y_a \cdot \delta/y_a = y'_b.\end{aligned}$$

- Non-negative constraint: this is true since $\delta \leq y_a$.

□

See Algorithm 12 for the aggregation procedure.

For convenience, we restate Lemma 4.3.4, and then we give a full proof.

Lemma 4.3.4 (restated). *$\forall j \in V$ whose opening moved from i' to i,*

- *k-center:* $d(i, j) \leq 5t$,
- *k-median:* $d(i, j) \leq 3d(i', j) + 8C_j$,
- *k-means:* $d(i, j)^2 \leq 15d(i', j)^2 + 80C_j$.

Algorithm 12 Aggregation procedure

Input: V, fractional (x, y), empires $\{\mathcal{E}_j\}$

For each $\mathcal{E}_u$

- Define $Y_u = \sum_{i \in \mathcal{E}_u} y_i$, $z_u = \frac{Y_u}{\lfloor Y_u \rfloor}$.
- **While** $\exists v$ s.t. $y_v \neq z_u$
  - Let v be the point farthest from u with nonzero y_v.
  - Let v' be the point closest to j with $y_{v'} \neq z_u$.
  - Move $\min\{y_v, z_u - y_{v'}\}$ units of opening from y_v to $y_{v'}$.

Output: updated (x, y)

Proof. k**-center.** Use the fact that all $C_j = t$, and $x_{ij} > 0 \implies d(i, j) \leq t$ with property (1b) to get:

$$\begin{aligned} d(i,j) &\leq d(i,u) + d(u,i') + d(i',j) \\ &\leq 2C_i + 2C_{i'} + d(i',j) \leq 5t. \end{aligned}$$

k**-median.** By construction, if the demand of point j moved from i' to i, then $\exists u \in \mathcal{M}$ s.t. $i, i' \in \mathcal{E}_u$ and $d(u, i) \leq d(u, i')$. Denote j' as the closest point in $\mathcal{M}$ to j. Then $d(u, i') \leq d(j', i')$ because $i' \in \mathcal{E}_u$. Then,

$$\begin{aligned} d(i,j) &\leq d(i,u) + d(u,i') + d(i',j) \\ &\leq 2d(u,i') + d(i',j) \\ &\leq 2d(j',i') + d(i',j) \\ &\leq 2(d(j',j) + d(j,i')) + d(i',j) \\ &\leq 8C_j + 3d(i',j). \end{aligned}$$

k**-means** The argument is similar to k-median, but with a bigger constant factor because of the squared triangle inequality.

$$\begin{aligned} d(i,j)^2 &\leq (d(i,u) + d(u,i') + d(i',j))^2 \\ &\leq (2d(u,i') + d(i',j))^2 \\ &\leq 4d(u,i')^2 + d(i',j)^2 + 4d(u,i')d(i',j) \\ &\leq 4d(u,i')^2 + d(i',j)^2 + 4d(u,i')d(i',j) + (2d(i',j) - d(u,i))^2 \\ &\leq 5d(u,i')^2 + 5d(i',j)^2 \\ &\leq 5d(j',i')^2 + 5d(i',j)^2 \\ &\leq 5(d(j',j) + d(j,i'))^2 + 5d(i',j)^2 \\ &\leq 5d(j',j)^2 + 10d(i',j)^2 + 10d(j',j)d(i',j) \\ &\leq 5d(j',j)^2 + 10d(i',j)^2 + 10d(j',j)d(i',j) + 5(d(j',j) - d(i',j))^2 \\ &\leq 10d(j',j)^2 + 15d(i',j)^2 \\ &\leq 80C_j + 15d(i',j)^2. \end{aligned}$$

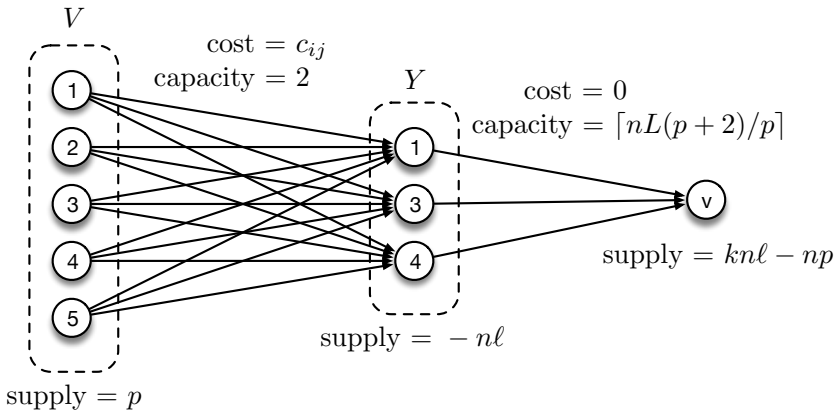


Figure 4.2: **Flow network for rounding the x's**: The nodes in each group all have the same supply, which is indicated below each group. The edge costs and capacities are shown above each group. The y-rounded solution gives a feasible flow in this network. By the Integral Flow Theorem, there exists a minimum cost flow which is integral and we can find it in polynomial time.

□

Step 4: Min cost flow Now we must round the x's. We set up a min cost flow problem, where an integral solution corresponds to an assignment of points to centers. We create a bipartite graph with V on the left (each with supply p) and the k centers on the right (each with demand $n\ell$), and a sink vertex with demand $np - kn\ell$. We carefully set the edge weights so that the minimum cost flow that satisfies the capacities corresponds to an optimal clustering assignment. See Figure 4.2.

Then using the Integral Flow Theorem, we are guaranteed there is an *integral* assignment that achieves the same optimal cost (and finding the min cost flow is a well-studied polynomial time problem [Papadimitriou and Steiglitz, 1998]). Thus, we can round the x's without incurring any additional cost to the approximation factor. This is the first time this technique has been used in the setting of clustering.

Step 4 details Set $\{i \mid y_i \neq 0\} = Y$. We show details of the min cost flow network in Algorithm 13.

Lemma 4.3.7. *There exists an integral assignment of the x'_{ij}'s such that $\forall i, j \in V$, $x'_{ij} \leq 2$ and it can be found in polynomial time.*

Proof. See Algorithm 13 and Figure 4.2 for the details of the flow construction.

In this graph, there exists a feasible flow: $\forall i, j \in V$, send x'_{ij} units of flow along the edge from i to j, and send $\sum_{j \in V} x_{ij}$ units of flow along the edge from i to v. Therefore, by the integral flow theorem, there exists a maximal integral flow which we can find in polynomial time. Also, by construction, this flow corresponds to an integral assignment of the x'_{ij}'s such that $x'_{ij} \leq 2$. □

Now we are ready to prove Theorem 4.3.1. The approximation ratios are 5, 11, and 95 for k-center, k-median, and k-means, respectively. For convenience, we restate the theorem statement.

Algorithm 13 Min cost flow procedure: Set up flow problem to round x's

Input: V, (x, y), y are integral

Create a flow graph $G = (V', E)$ as follows.

- Add each $i \in V$ to V', and give i supply p.
- Add each $i \in Y$ to V', and give i demand $n\ell$.
- Add a directed edge (i, j) for each $i \in V$, $j \in Y$, with capacity 2 and cost c_{ij} (for k-center, make the edge weight $5t$ if $d(i, j) \leq 5t$ and $+\infty$ otherwise.
- Add a sink vertex v to V', with demand $np - kn\ell$.
- Add a directed edge (i, v) for each $i \in Y$, with capacity $\lceil \frac{p+2}{p} nL \rceil - n\ell$ and cost 0.

Run an min cost integral flow solver on G.

Update x by setting x_{ij} to 0, 1, or 2 based on the amount of flow going from i to j.

Output: updated (x, y) with integral x's and y's

Theorem 4.3.1 (restated). *Algorithm 4.1 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.*

Proof of Theorem 4.3.1.

k-center: Recall that we defined $C_{LP} = tnp$, where t is the threshold for the k-center LP. From Lemma 4.3.4, when we reassign the demand of point j from i' to i, $d(i, j) \leq 5t$. In other words, the y-rounded solution is feasible at threshold $5t$. Then the k-center cost of the new y's is $np(5t) = 5C_{LP}$. From Lemma 4.3.7, we can also round the x's at no additional cost.

k-median: From Property 4.3.4, when we reassign the demand of point j from i' to i, $d(i, j) \leq 3d(i', j) + 8C_j$. Then we can bound the cost of the new assignments with respect to the original LP solution as follows.

$$\begin{aligned}
\sum_{i \in V} \sum_{j \in V} d(i, j) x'_{ij} &\leq \sum_{i \in V} \sum_{j \in V} (8C_j + 3d(i, j)) x_{ij} \\
&\leq \sum_{i \in V} \sum_{j \in V} 8C_j x_{ij} + \sum_{i \in V} \sum_{j \in V} 3d(i, j) x_{ij} \\
&\leq \sum_{j \in V} 8C_j \sum_{i \in V} x_{ij} + 3C_{LP} \\
&\leq \sum_{j \in V} 8pC_j + 3C_{LP} \\
&\leq 11C_{LP}.
\end{aligned}$$

Then from Lemma 4.3.7, we get a solution of cost at most $11C_{LP}$, which also has integral x's.

k-means: The proof is similar to the k-median proof. From lemma 4.3.4, when we reassign the demand of point j from i' to i, $d(i, j)^2 \leq 15d(i', j)^2 + 80C_j$. Then we can bound the cost of

the new assignments with respect to the original LP solution as follows.

$$\begin{aligned}
\sum_{i\in V}\sum_{j\in V} d(i,j)^2 x'_{ij} &\leq \sum_{i\in V}\sum_{j\in V}(80C_j + 15d(i',j)^2)x_{ij} \\
&\leq \sum_{i\in V}\sum_{j\in V} 80C_j x_{ij} + \sum_{i\in V}\sum_{j\in V} 15d(i,j)^2 x_{ij} \\
&\leq \sum_{j\in V} 80C_j \sum_{i\in V} x_{ij} + 15C_{LP} \\
&\leq \sum_{j\in V} 80pC_j + 15C_{LP} \\
&\leq 95C_{LP}.
\end{aligned}$$

Then from Lemma 4.3.7, we get a solution of cost at most $95C_{LP}$, which also has integral x's. □

See Algorithm 14 for the final algorithm.

Algorithm 14 Bicriteria approximation Algorithm for k-median, k-means, and k-center

Input: V

- Run a solver for the LP relaxation for k-median, k-means, or k-center, output (x, y).
- Run Algorithm 11 with V, (x, y), output set of empires $\{\mathcal{E}_j\}$.
- Run Algorithm 12 with V, $\{\mathcal{E}_j\}$, (x, y), output updated (x, y).
- Run Algorithm 13 with V, (x, y), output updated (x, y).

Output: Integral (x, y) corresponding to bicriteria clustering solution

In the next section, we show a more involved algorithm specifically for k-center which achieves a 6-approximation with *no violation* to the capacity or replication constraints.

4.3.2 True approximation algorithm for k-center

In this section, we present a more complicated algorithm that is specific to k-center, which achieves a true approximation algorithm - the capacity and replication constraints are no longer violated.

As in the previous section and in prior work [An et al., 2014, Cygan et al., 2012, Khuller and Sussmann, 1996], we start off by guessing the optimal distance t. Since there are a polynomial number of possibilities, it is still only polynomially expensive. We then construct the threshold graph $G_t = (V, E_t)$, with j being the set of all points, and $(x, y) \in E_t$ iff $d(x, y) \leq t$.

A high-level overview of the rounding algorithm that follows is given in Algorithm 15.

Connection to the previous section The algorithm here is similar to the bicriteria algorithm presented previously. There are, however, two differences. Firstly, we work only with connected components of the threshold graph. This is necessary to circumvent the unbounded integrality gap of the LP [Cygan et al., 2012]. Secondly, the rounding procedure of the y's can now move opening

across different empires. Since the threshold graph is connected, the distance between any two adjacent monarchs is bounded and turns out to exactly be thrice the threshold. This enables us to get a constant factor approximation without violating any constraints.

Algorithm 15 Algorithm overview

Input: V: the set of points, k: the number of clusters, (ℓ, L): min and max allowed cluster size

Procedure *balanced-k-center*(V, k, p, ℓ, L)

- **For each** threshold t
  - Construct the threshold graph $G_t = (V, E_t)$, where $E_t = \{(u, v) \mid d(u, v) \leq t\}$
  - **For each** connected component $G^{(c)}$ of G_t and each $1 \leq k' \leq k$,
    - Solve *LPRound*$(G^{(c)}, k', p, \ell, L)$
  - Search for a solution for each $G^{(c)}$ with k_c centers such that $\sum_c k_c = k$ by linear search, call it s
  - If $s \neq \emptyset$, **return** s.

Procedure *LPRound*(G, k, p, ℓ, L)

- $(x, y) \leftarrow$ relaxed solution of LP in equation 4.3
- *Round the y values*
  - Run Algorithm 16 on (x, y) to return a tree T
  - Run Algorithm 17) to return (x', y') where y' is integral
- Round x' to get x'' from Theorem 4.3.16
- **Return** (x'', y')

Output: A k-clustering of V respecting cluster size constraints, p: replication factor

The Algorithm

Intuition

The approach is to guess the optimal threshold, construct the threshold graph at this threshold, write and round several LPs for each connected component of this graph for different values of k. The intuition behind why this works is that at the optimal threshold, each cluster is fully contained within a connected component (by definition of the threshold graph).

We the round the opening variables, but this time, open exactly k centers. Most of the work goes into rounding the openings, and showing that it is correct. Then, we simply round the assignments using a minimum cost flow again.

Linear Program

As earlier, let y_i be an indicator variable to denote whether vertex i is a center, and x_{ij} be indicators for whether j belongs to the cluster centered at i. By convention, i is called a facility and j is called a client.

Consider the following LP relaxation for the IP for each connected component of G. Note that it is exactly the same as the one from the previous section, except it is described in terms of the threshold graph G. Let us call it $\texttt{LP}\text{-}k\text{-}\texttt{center}(G)$:

$$\sum_{i \in V} y_i = k \tag{4.3a}$$

$$x_{ij} \leq y_i \qquad \forall i, j \in V \tag{4.3b}$$

$$\sum_{j : ij \in E} x_{ij} \leq nLy_i \qquad \forall i \in V \tag{4.3c}$$

$$\sum_{j : ij \in E} x_{ij} \geq n\ell y_i \qquad \forall i \in V \tag{4.3d}$$

$$\sum_{i : ij \in E} x_{ij} = p \qquad \forall j \in V \tag{4.3e}$$

$$x_{ij} = 0 \qquad \forall ij \notin E \tag{4.3f}$$

$$0 \leq x, y \leq 1 \tag{4.3g}$$

Once we have the threshold graph, for the purpose of k-center, all distances can now be measured in terms of the length of the shortest path in the threshold graph. Let $d_G(i, j)$ represent the distance between i and j measured by the length of the shortest path between i and j in G.

Connected Components

It is well known [Cygan et al., 2012] that even without lower bounds and replication, the LP has unbounded integrality gap for general graphs. However, for connected components of the threshold graph, this is not the case.

To begin with, we show that it suffices to be able to do the LP rounding procedure for only connected threshold graphs, even in our generalization.

Theorem 4.3.8. *If there exists an algorithm that takes as input a connected graph G, capacities ℓ, L, replication p, and k for which $\texttt{LP}\text{-}k\text{-}\texttt{center}(G_t)$ is feasible, and computes a set of k centers to open and an assignment of every vertex j to p centers i such that $d_G(i, j) \leq r$ satisfying the capacity constraints, then we can obtain a r-approximation algorithm to the balanced k-centers problem with p-replication.*

Proof. Let connected component i have k_i clusters. For each connected component, do a linear search on the range $[1, \ldots, k]$ to find values of k_i for which the problem is feasible. These feasible values will form a range, if size constraints are to be satisfied. To see why this is the case, note that if (x_1, y_1) and (x_2, y_2) are fractional solutions for $k = k_1$ and $k = k_2$ respectively, then $((x_1 + x_2)/2, (y_1 + y_2)/2)$ is a valid fractional soluion for $k = (k_1 + k_2)/2$.

Suppose the feasible values of k_i are $m_i \leq k_i \leq M_i$. If $\sum_i m_i > k$ or $\sum_i M_i < k$, return NO (at this threshold t). Otherwise, start with each k_i equal to m_i. Increase them one by one up to M_i until $\sum_i k_i = k$. This process takes polynomial time. □

From now on, the focus is entirely on a single connected component.

Rounding y

Given an integer feasible point to the IP for each connected component, we can obtain the desired clustering. Hence, we must find a way to obtain an integer feasible point from any feasible point of LP-k-center.

To round the y, we follow the approach of An et al. [2014]. The basic idea is to create a coarse clustering of vertices, and have the cluster centers form a tree. The radius of each cluster will be at most 2, and the length of any edge in the tree will exactly be three, by construction.

Now, to round the y, we first start from the leaves of the tree, moving opening around in each coarse cluster such that at most one node (which we pick to be the center, also called the monarch). In subsequent steps, this fractional opening is passed to the parent cluster, where the same process happens. The key to getting a constant factor approximation is to ensure that fractional openings that transferred from a child cluster to a parent cluster are not propagated further. Note that the bicriteria algorithm did not move opening from one coarse cluster (empire) to another because we didn't have an upper bound of the cost incurred by making this shift.

Preliminaries. We start with some definitions.

Definition 4.3.9 (δ-feasible solution [Cygan et al., 2012])**.** *A solution (x, y) feasible on G^δ, the graph obtained by connecting all nodes within δ hops away from each other.*

Next, we introduce the notion of a distance-r shift. Intuitively, a distance-r shift is a series of movements of openings, none of which traverses a distance more than r in the threshold graph. Note that the definition is similar to what is used in An et al. [2014].

Definition 4.3.10 (Distance-r shift)**.** *Given a graph $G = (V, E)$ and $y, y' \in \mathbb{R}^{|V|}_{\geq 0}$, y' is a distance-r shift of y if y' can be obtained from y via a series of disjoint movements of the form "Move δ from i to i'" where $\delta \leq \min(y_i, 1 - y_{i'})$ and every i and i' are at most a distance r apart in the threshold graph G. Further, if all y' are zero or one, it is called an integral distance-r shift.*

Note that, by the definition of a distance-r shift, each unit of y moves only once and if it moves more than once, all the movements are put together as a single, big movement, and this distance still does not exceed r.

Lemma 4.3.11 (Realizing distance-r shift)**.** *For every distance-r shift y' of y such that $0 \leq y'_i \leq 1 \; \forall i \in V$, we can find x' in polynomial time such that (x', y') is $(r + 1)$-feasible.*

Proof. We can use the Move operation described earlier and in Cygan et al. [2012] to change the corresponding x for each such a movement to ensure that the resulting (x', y') are $(r + 1)$-feasible. The additional restriction $\delta \leq 1 - y_b$ ensures that $y \leq 1$. Since each unit of y moves only once, all the movements put together will also lead a solution feasible in G^{r+1}, i.e. we get a $(r + 1)$-feasible solution.

□

From here on, we assume that $x_{ij}, x_{i'j}$ are adjusted as described above for every movement between i and i'.

The algorithm to round y [An et al., 2014] proceeds in two phases. In the first phase, we cluster points into a tree of coarse clusters (monarchs) such that nearby clusters are connected using the monarch procedure of Khuller and Sussmann [1996]. In the second phase, fractional opening are aggregated to get an integral distance-5 shift.

Monarch Procedure. The monarch procedure is presented a little differently but is very similar to the monarch procedure presented earlier. Since the threshold graph is connected, we can get guarantees on how big the distance between two monarchs is.

Algorithm 16 describes the first phase where we construct a tree of monarchs and assign empires to each monarch. Let $\mathcal{M}$ be the set of all monarchs. For some monarch, $u \in \mathcal{M}$, let $\mathcal{E}_u$ denote its empire. For each vertex i, let $m(i)$ denote the the monarch u to whose empire $\mathcal{E}_u$, i belongs.

Algorithm 16 Monarch Procedure: Algorithm to construct tree of monarchs and assign empires

Input: $G = (V, E)$

$\texttt{Marked} \leftarrow \emptyset$

For each $j \in V$

- initialize $\texttt{ChildMonarchs}(j)$ and $\texttt{Dependents}(j)$ to $\emptyset$

Pick any vertex u and make it a monarch

$\mathcal{E}_u \leftarrow N^+(u)$; Initialize T to be a singleton node u

$\texttt{Marked} \leftarrow \texttt{Marked} \cup \mathcal{E}_u$

While $\exists w \in (V \setminus \texttt{Marked})$ such that $d_G(w, \texttt{Marked}) \geq 2$

- Let $u \in (V \setminus \texttt{Marked})$ and $v \in \texttt{Marked}$ such that $d_G(u, v) = 2$
- Make u a monarch and assign its empire to be $\mathcal{E}_u \leftarrow N^+(u)$
- $\texttt{Marked} \leftarrow \texttt{Marked} \cup \mathcal{E}_u$
- Make u a child of $m(v)$ in T
- $\texttt{ChildMonarchs}(v) \leftarrow \texttt{ChildMonarchs}(v) \cup \{u\}$

For each $v \in (V \setminus \texttt{Marked})$

- Let $u \in \texttt{Marked}$ be such that $d_G(u, v)$ = 1
- $\texttt{Dependents}(u) \leftarrow \texttt{Dependents}(u) \cup \{v\}$
- $\mathcal{E}_{m(u)} \leftarrow \mathcal{E}_{m(u)} \cup \{v\}$

Output: Tree of monarchs, $T = (\mathcal{M}, E')$, and empires for each monarch

The guarantees now translate to the following (Lemma 4.3.12):

- Empires partition the point set.
- The empire includes *all* immediate neighbors of a monarch and additionally, some other nodes of distance two (called dependents).
- Adjacent monarchs are exactly distance 3 from each other.

Lemma 4.3.12. *Algorithm 16, the monarch procedure is well-defined and its output satisfies the following:*

- $\mathcal{E}_u \cap \mathcal{E}_{u'} = \emptyset$.
- $\forall u \in \mathcal{M} : \mathcal{E}_u = N^+(u) \cup (\bigcup_{j \in N^+(u)} \texttt{Dependents}(j))$.
- *The distance between a monarch and any node in its empire is at most 2.*
- *Distance between any two monarchs adjacent in T is exactly 3.*
- *If* $\texttt{ChildMonarchs}(j) \neq \emptyset$ *or* $\texttt{Dependents}(j) \neq \emptyset$*, then j is at distance one from some monarch.*

Proof. Note that the whole graph is connected and $V \neq \emptyset$. For the while loop, if there exists w such that $d_G(w, \texttt{Marked}) \geq 2$, there exists u such that $d_G(u, \texttt{Marked}) = 2$ because the graph is connected. By the end of the while loop, there are no vertices at a distance 2 or more from Marked. Hence, vertices not in Marked, if any, should be at a distance 1 from Marked. Thus, the algorithm is well defined.

Each time a new monarch u is created, $N^+(u)$ is added to its empire. This shows the first statement. The only other vertices added to any empire are the dependents in the foreach loop. Each dependent j is directly connected to i, a marked vertex. Hence, i has to be a neighbor of a monarch. If i were a monarch, j would have been marked in the while loop. Thus, $d_G(j, m(i)) = 2$.

If the first statement of the while loop, v is a marked vertex, and has to be a neighbor of some monarch $m(v)$. New monarch u is chosen such that $d_G(u, v) = 2$. The parent monarch of u is $m(v)$ and $d_G(u, m(v)) = d_G(u, v) + d_G(v, m(v)) = 3$.

□

Initial Aggregation. Now, we shall turn to the rounding algorithm of An et al. [2014]. The algorithm begins with changing y_u of every monarch $u \in \mathcal{M}$ to 1. Call this the initial aggregation. It requires transfer of at most distance one because the neighbors of the monarchs has enough opening.

Lemma 4.3.13. *The initial aggregation can be implemented by a distance-1 shift.*

Proof. For every vertex $u \in V$, we have $\sum_{j \in N(u)} y_j \geq \sum_{j \in N(u)} x_{uj} = p \geq 1$. Hence, there is enough y-mass within a distance of one from u. The actual transfer can happen by letting $\delta = \min(1 - y_u, y_j)$ for some neighbor j of u and then transferring δ from j to u. That is, $y_j = y_j - \delta$ and $y_u = y_u + \delta$.

□

Rounding. The rounding procedure now proceeds in a bottom-up manner on the tree of monarchs, rounding all y using movements of distance 5 or smaller. After rounding the leaf empires, all fractional opening, if any is at the monarch. For internal empires, the centers of child monarch (remnants of previous rounding steps) and dependents are first rounded. Then the neighbors of the

monarch are rounded to leave the entire cluster integral except the monarch. The two step procedure is adopted so that the opening propagated from this monarch to its parent originates entirely from the 1-neighborhood of the monarch.

Formally, at the end of each run of round on $u \in \mathcal{M}$, all the vertices of the set I_u are integral, where $I_u := (\mathcal{E}_u \setminus u) \cup (\bigcup_{j \in N(u)} \texttt{ChildMonarchs}(j))$.

Algorithm 17 Algorithm to round y

Input: Tree of monarchs, T, and empires for each monarch after the initial aggregation

Procedure *Round*(Monarch u)

- **For each** child w of u in T, *Round*(w) //Recursive call
- **For each** $j \in N(u)$ //Phase 1
  - $X_j \leftarrow \{j\} \cup \texttt{ChildMonarchs}(j) \cup \texttt{Dependents}(j)$
  - $W_j \leftarrow \{\lfloor y(X_j) \rfloor$ nodes from $X_j\}$; (Avoid picking j if possible)
  - Run *LocalRound*$(W_j, X_j, \emptyset)$
  - Run *LocalRound*$(\{j\}, X_j \setminus W_j, \emptyset)$
- $F = \{j | j \in N(u)$ and $0 < y_j < 1\}$ //Phase 2
- $W_F \leftarrow \{$ any $\lfloor y(F) \rfloor$ nodes from $F\}$
- Run *LocalRound*$(W_F, F, \emptyset)$
- **If** $y(F \setminus W_F) > 0$
  - Choose $w^* \in F \setminus W_F$
  - Run *LocalRound*$(\{w^*\}, F \setminus W_F, u)$

Procedure *LocalRound*(V_1, V_2, V_3)

- **While** $\exists i \in V_1$ such that $y_i < 1$
  - Choose a vertex w with non-zero opening from $V_2 \setminus V_1$
  - If there exists none, choose j from $V_3 \setminus V_1$
  - $\delta \leftarrow \min(1 - y_i, y_j)$
  - `Move` δ from j to i

Output: y', an integral distance-5 shift of y

The rounding procedure is described in detail in Algorithm 17. The following lemma states and proves that algorithm 17 rounds all points and doesn't move opening very far.

Lemma 4.3.14 (Adaptation of Lemma 19 of An et al. [2014])**.** *Let*

$$I_u := (\mathcal{E}_u \setminus u) \cup (\bigcup_{j \in N(u)} \texttt{ChildMonarchs}(j)).$$

- `Round`(u) *makes the vertices of* I_u *integral with a set of opening movements within* $I_u \cup \{u\}$.
- *This happens with no incoming movements to the monarch* u *after the initial aggregation.*

- *The maximum distance of these movements is five, taking the initial aggregation into account.*

Proof. **Integrality.** From lemma 4.3.12, it can be seen that $X_j, j \in N(u)$ above form a partition of I_u. Hence, it suffices to verify that each node of every X_j is integral.

At the end of line 1, the total non-integral opening in X_j is $y(X_j) - \lfloor y(X_j) \rfloor$, and is hence smaller than one. Line 1 moves all these fractional openings to j. By now, all openings of $X_j \setminus \{j\}$ are integral.

Now, F is the set of all non-integral $j \in N(u)$. So, by the end of line 1, the total non-integral opening in $N(u)$ (and hence in all of I_u) is $y(F \setminus W_F) = y(F) - \lfloor y(F) \rfloor$, and is again smaller than one. If this is zero, we are done.

Otherwise, we choose a node w^*, shift this amount to w^* in line 1. To make this integral, this operations also transfers the *remaining amount, i.e.* $1 - y(F \setminus W_F)$ *from the monarch* u. If this happens, the monarch u's opening is no longer integral, but I_u's is.

This shows the first bullet. For the second one, notice that after the initial aggregation, this last operation is the only one involving the monarch u and hence, there are no other incoming movements into u.

Distance. In the first set of transfers in line 1 the distance of the transfer is at most 4. This is because dependents are a distance one away from j and child monarchs are at a distance two away. The maximum distance is when the transfer happens from one child monarch to another, and this distance is 4 (recall that there are no incoming movements into monarchs).

The transfers in line 1 moves openings from a child monarch or a dependent to j. The distances are 2 and 1 respectively. Accounting for the initial aggregation, this is at most 3.

The rounding on line 1 moves openings between neighbors of the monarch, i.e. from some j to j' where $j, j' \in N(u)$. So, the distance between j and j' is at most 2. From the preceding transfers, the openings at j moved a distance of at most three to get there, and thus, we conclude that openings have moved at most a distance of 5 so far.

The first step of rounding on line 1 moves openings from some j to w^*, where $j, w^* \in N(u)$. As above, the maximum distance in this case is 5. The second step of rounding on line 1 moves opening from the monarch u to its neighbor w^*. This distance is one, and after accounting for the initial aggregation, is 2.

From this, we see that the maximum distance any opening has to move is 5. □

The algorithms, their properties in conjunction with lemma 4.3.11 leads to the following theorem, which also summarizes this subsection.

Theorem 4.3.15. *There exists a polynomial time algorithm to find a 6-feasible solution with all y integral.*

Rounding x

Once we have integral y, rounding the x is fairly straight-forward, without making the approximation factor any worse. Exactly the same procedure used in bicriteria algorithms works here

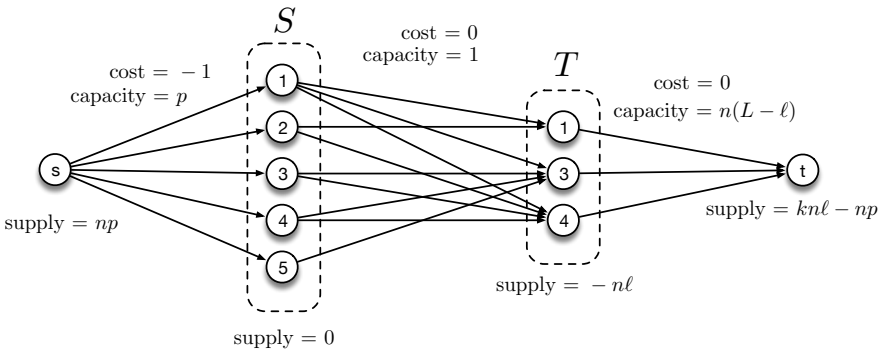


Figure 4.3: Minimum cost flow network to round x's. Each node in a group has the same supply, which is indicated below. The cost and capacity of each edge is indicated above.

too. But, we can have an easier construction since for k-center since we can use distances in the threshold graph instead.

Theorem 4.3.16. *There exists a polynomial time algorithm that given a δ-feasible solution (x, y) with all y integral, finds a δ-feasible solution (x', y) with all x' integral.*

Proof. We shall use a minimum cost flow network to this. Consider a directed bipartite graph (S, T, E'), where $S = V$ and $T = \{i : y_i = 1\}$ and $j \to i \in E'$ iff $x_{ij} > 0$. Add a dummy vertex s, with edges to every vertex in S, and t with edges from every vertex in T. In this network, let every edge of the bipartite graph have capacity 1. Further, all the $s \to S$ edges have capacity p. s supplies a flow of np units, while each $u \in T$ has a demand of l units. To ensure no excess demand or supply, t has a demand of $np - kl$. All the $t \to T$ edges have a capacity of $(L - \ell)$.

All the $s \to S$ edges have a cost of -1 and every other edge has a cost of zero. See figure 4.3.

Clearly, a feasible assignment (x, y) to $\texttt{LP-}k\texttt{-center}(G^\delta)$ with integral y is a feasible flow in this network. In fact, it is a minimum-cost flow in this network. This can be verified by the absence of negative cost cycles in the residual graph (because all negative cost edges are at full capacities).

Since, the edge capacities are all integers, there exists a minimum cost integral flow by the Integral Flow Theorem. This flow can be used to fix the cluster assignments.

□

Piecing together theorems 4.3.15 and 4.3.16, we have the following theorem:

Theorem 4.3.17. *Given an instance of the k-centers problem with p-replication and for a connected graph G, and a fractional feasible solution to $\texttt{LP-}k\texttt{-center}(G)$, there exists a polynomial time algorithm to obtain a 6-feasible integral solution. That is, for every i, j such that $x_{ij} \neq 0$, we have $d_G(i, j) \leq 6$.*

4.4 Structure of Balanced Clustering

In this section, we show that adding lower bounds to clustering makes the problem highly non-trivial. Specifically, our main result is that the k-means, k-median, and k-center objective values

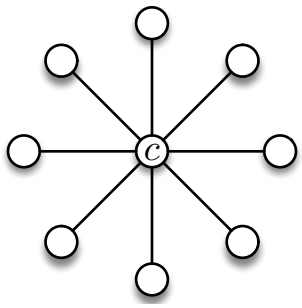


Figure 4.4: A graph in which the objective function strictly increases with k. Each edge signifies distance 1, and all other distances are 2.

may oscillate arbitrarily with respect to k (Theorem 4.4.4). In light of this structure, our results from Section 4.3 are more surprising, since it is not obvious that algorithms with constant-factor guarantees exist.

We give a variety of clustering instances which do not have monotone cost functions with respect to k. For readability and intuition, these examples start out simple, and grow in complexity until we eventually prove Theorem 4.4.4.

First, consider a star graph with n points and lower bound ℓ, such that $n\ell \geq 3$ (see Figure 4.4). The center c is at distance 1 to the $10n\ell$ leaves, and the leaves are at distance 2 from each other. When $k = 1$, each point is distance 1 to the center c. However as we increase k, the new centers must be leaves, distance 2 from all the other points, so $n\ell - 1$ points must pay 2 instead of 1 for each extra center. It is also easy to achieve an objective that strictly decreases up to a local minimum k', and then strictly increases onward, by adding k' copies of the center of the star.

Lemma 4.4.1. *Given a star graph with parameters n and ℓ such that $n\ell \geq 3$, then the cost of the k-means and k-median objectives strictly increase in k.*

Proof. Let the size of the star graph be n. Clearly, the optimal center for $k = 1$ is c. Then $\mathcal{OPT}_1 = n - 1$. Then for $k = 2$, we must choose another center p that is not c. p is distance 2 to all points other than c, so the optimal clustering is for p's cluster to have the minimum of $n\ell$ points, and c's cluster has the rest. Therefore, $\mathcal{OPT}_2 = n + n\ell - 2$.

This process continues; every time a new center is added, the new center pays 0 instead of 1, but $n\ell - 1$ new points must pay 2 instead of 1. This increases the objective by $n\ell - 2$. As long as $n\ell \geq 3$, this ensures the objective function is strictly increasing in k. □

Note for this example, the problem goes away if we are allowed to place multiple centers on a single point (in the literature, this is called "soft capacities", as opposed to enforcing one center per point, called "hard capacities"). The next lemma shows there can be a local minimum for hard capacities.

Lemma 4.4.2. *For all k', there exists a balanced clustering instance in which the k-means or k-median objective as a function of k has a local minimum at k'.*

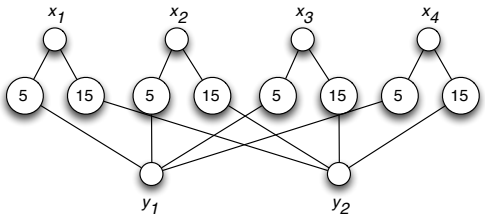


Figure 4.5: Each edge signifies distance 1, and all other distances are 2. The middle points are replicated as many times as their label suggests (but each pair of replicated points are still distance 2 away). Finally, add length 1 edges between all pairs in $\{x_1, x_2, x_3, x_4\}, \{y_1, y_2\}$.

Proof. Given $l \geq 3$, we create a clustering instance as follows. Define k' sets of points $G_1, \ldots, G_{k'}$, each of size $2n\ell - 1$. For any two points in some G_i, set their distance to 0. For any two points in different sets, set their distance to 1. Then for $1 \leq k \leq k'$, the objective value is equal to $(k' - k)(2n\ell - 1)$, since we can put k centers into k distinct groups, but $(k' - k)$ groups will not have centers, incurring cost $2n\ell - 1$. When $k > k'$, we cannot put each center in a distinct group, so there is some group G_i with two centers. Since $|G_i| = 2n\ell - 1$, the two centers cannot satisfy the capacity constraint with points only from G_i, so the objective value increases. □

Local maxima So far, we have seen examples in which the objective decreases with k, until it hits a minimum (where capacities start to become violated), and then the objective strictly increases. The next natural question to ask, is whether the objective can also have a local maximum. We show the answer is yes in the following lemma.

Lemma 4.4.3. *There exists a balanced clustering instance in which the k-center, k-median, and k-means objectives contain a local maximum with respect to k.*

Proof. Consider Figure 4.5, where $n = 86$, and set $n\ell = 21$. First we give the intuition behind the proof, and later we provide the full details. Since the distances are all 1 or 2, this construction is trivially a valid distance metric. From Figure 4.5, we see that $k = 2$ and $k = 4$ have valid clusterings using only length 1 edges, using centers $\{y_1, y_2\}$ and $\{x_1, x_2, x_3, x_4\}$, respectively. But now consider $k = 3$. The crucial property is that by construction, y_1 and any x_i cannot simultaneously be centers and each satisfy the capacity to distance 1 points, because the union of their distance 1 neighborhoods is less than $2n\ell$. Then we carefully check all other sets of 3 centers do not achieve a clustering with distance 1 edges, which completes the proof.

Here are the full details. Consider the graph in Figure 4.5, where $n = 86$, and set $n\ell = 21$. Since the distances are all 1 or 2, this construction is trivially a valid distance metric. From Figure 4.5, we see that $k = 2$ and $k = 4$ have valid clusterings using only length 1 edges, using centers $\{y_1, y_2\}$ and $\{x_1, x_2, x_3, x_4\}$, respectively. But now consider $k = 3$. The crucial property is that by construction, y_1 and any x_i cannot simultaneously be centers and each satisfy the capacity to distance 1 points, because the union of their distance 1 neighborhoods is less than $2n\ell$. So we cannot just take the centers from $k = 2$ and add a center from $k = 4$. Formally, we show no possible set of 3 centers can be distance 1 to all points without violating the lower bound on the cluster sizes.

Case 1: the set of centers includes a point p not in $\{x_1, x_2, x_3, x_4, y_1, y_2\}$. The rest of the points are only distance 1 from exactly two points, so p cannot hit the lower bound of 21 using only distance 1 assignments.

Case 2: the set of centers is a subset of $\{x_1, x_2, x_3, x_4\}$. Then there are clearly 20 points which are not distance 1 from the three centers.

Case 3: the set of centers includes both y_1 and y_2. Then we need to pick one more center, x_i. x_i is distance 1 from 20 middle points, plus $\{x_1, x_2, x_3, x_4, y_1, y_2\}$, so 26 total. y_1 is also distance 1 from 20 middle points and $\{x_1, x_2, x_3, x_4, y_1, y_2\}$. y_1 and x_i share exactly 5 neighbors from the middle points, plus $\{x_1, x_2, x_3, x_4, y_1, y_2\}$ as neighbors. Then the union of points that x_i and y_1 are distance 1 from, is $26 + 26 - 11 = 41$, which implies that x_i and y_1 cannot simultaneously reach the lower bound of 21 with only distance 1 points.

Case 4: the set of centers does not include x_i nor y_j. By construction, for each pair x_i and y_j, there exists some middle points which are only distance 1 from x_i and y_j.

These cases are exhaustive, so we conclude $\mathcal{OPT}_3$ must be strictly larger than $\mathcal{OPT}_2$ and $\mathcal{OPT}_4$ (no matter what objective we use). □

The previous example does not work for the case of soft capacities, since the set of centers $\{x_1, y_2, y_2\}$ allows every point to have an edge to its center. Now we prove our main theorem. Note, this theorem holds even for soft capacities.

Theorem 4.4.4. *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.*

Proof. First we give a sketch of the proof, focusing on intuition, and then we give the full details. As in the previous lemma, we will construct a set of points in which each pair of points are either distance 1 or 2. It is convenient to define a graph on the set of points, in which an edge signifies a distance of 1, and all non-edges denote distance 2. We will construct 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 $12m$.

We start by 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 with the following properties.

1. For each even value of k between $10m$ and $12m$, there is an assignment of the points in Y to the centers in X_k so that points in Y are only assigned to adjacent centers and the capacity constraints are satisfied.

2. Each of the good centers in X is adjacent to no more than $\frac{6}{5} \cdot n\ell$ points in Y.

3. For each good center x in X_k, there is at least one point x' in every other set X'_k (for $k' \neq k$) so that the number of points in Y adjacent to both x and x' is at least $\frac{2}{5} \cdot n\ell$.

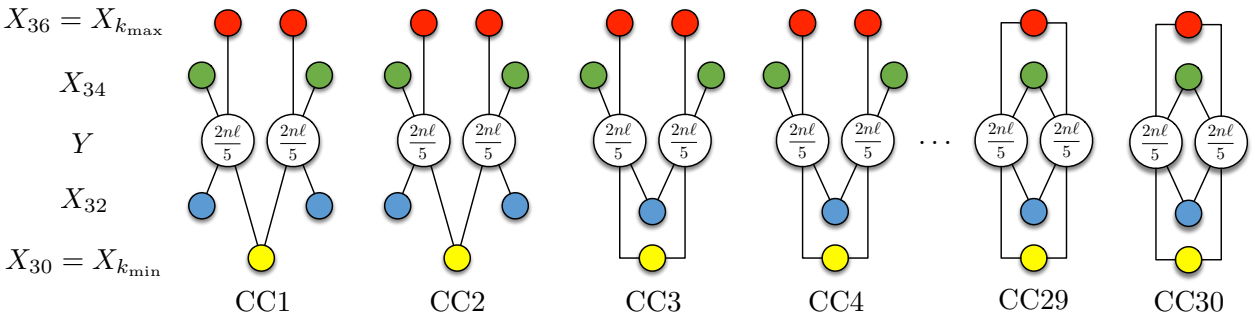


Figure 4.6: An example when $m = 3$. Each X_k is a different color. Each edge signifies distance 1, and all other distances are 2. The middle points are replicated as many times as their label suggests (but each pair of replicated points are still distance 2 away).

4. Any subset of the centers in X that does not contain any complete set of good centers X_k for some even k is non-adjacent to at least one point in Y.

Whenever we add a point to Y, we give it an edge to exactly one point from each X_k. This ensures that each X_k partitions Y. We first create connected components as in Figure 4.6 that each share $\frac{2}{5} \cdot n\ell$ points from Y, to satisfy Property 3.

For property 4, we add one additional point to Y for every combination of picking one point from each X_k. This ensures that any set which does not contain at least one point from each X_k will not be a valid partition for Y. Note that in the previous two steps, we did not give a single center more than $\frac{6}{5} \cdot n\ell$ edges, satisfying property 2. Then we add "filler" points to bring every center's capacity up to at least $n\ell$, which satisfies property 1.

Now we explain why properties 1-4 are sufficient to finish off the proof. Property 1 guarantees that the for each even value of k there is a clustering where the cost of each point in Y is one, which results in a good clustering objective.

Properties 2 and 3 guarantee that any set including a full X_k and a point from a different $X_{k'}$ cannot achieve cost 1 for each point without violating the capacities. Property 4 guarantees that any set without a full X_k cannot achieve cost 1 for each point. This completes the proof sketch.

Now we give the full details of the proof.

Setup. Set $k_{min} = 10 \cdot m$, and $k_{max} = 12m$. Define $K_{good} = \{k \mid k_{min} \leq k \leq k_{max} \text{ and } 2 \mid k\}$. Similarly, let $K_{bad} = \{k \mid k_{min} \leq k \leq k_{max} \text{ and } 2 \nmid k\}$. Note $|K_{bad}| = m$ and $|K_{good}| = m + 1$. For all $k \in K_{good}$, define $X_k = \{x_1^{(k)}, \ldots, x_{k'}^{(k)}\}$. Let $X = \bigcup_k X_k$.

Define $G = (V, E)$, $V = X \cup Y$, $X \cap Y = \emptyset$. Just like in the last proof, the edges later correspond to a distance of 1, and all other distances are 2. We will construct Y and E such that for all $k \in K_{good}$, all the neighbors of X_k form a partition of Y, i.e. $\forall k \in K_{good}$, $\bigcup_i N(x_i^{(k)}) = Y$ and $N(x_i^{(k)}) \cap N(x_j^{(k)}) = \emptyset$ for all $i \neq j$. So taking X_k as the centers corresponds to a k-clustering in which all points are distance 1 from their center. We will also show that for all $k \in K_{bad}$, it is not possible to find a valid set of centers for which every point has an edge to its center, unless the capacities are violated. This implies that all m points in K_{bad} are local maxima.

For all $k \in K_{good}$, $X_{k'}$ will have exactly $\frac{k_{max}}{k'} l$ edges in Y Thus, set $n\ell = \prod_{k \in K_{good}} k$ to make all of these values integral. Note that some points (those in $X_{k_{max}}$) have exactly $n\ell$ edges, and all points have $\leq \frac{6}{5} n\ell$ edges (which is tight for the points in $X_{k_{min}}$).

Now we define the main property which drives the proof. We say $x_{i_1}^{(j_1)}$ *overlaps* with $x_{i_2}^{(j_2)}$ if $N(x_{i_1}^{(j_1)}) \cup N(x_{i_2}^{(j_2)}) > \frac{2}{5} n\ell$. Note this immediately implies it is not possible to include them in the same set of centers such that each point has an edge to its center, since $N(x_{i_1}^{(j_1)}) \cup N(x_{i_2}^{(j_2)}) \leq N(x_{i_1}^{(j_1)}) + N(x_{i_2}^{(j_2)}) - N(x_{i_1}^{(j_1)}) \cap N(x_{i_2}^{(j_2)}) < 2 \cdot \frac{6}{5} n\ell - \frac{2}{5} n\ell = 2n\ell$.

Outline. We will construct Y in three phases. First, we add edges to ensure that for all $x_{i_1}^{(j_1)}$, for all $j_2 \neq j_1$, there exists an i_2 such that $x_{i_1}^{(j_1)}$ overlaps with $x_{i_2}^{(j_2)}$. It follows that if we are trying to construct a set of centers from X for $k' \in K_{bad}$, we will not be able to use any complete $X_{k'}$ as a subset. These are called the *backbone* edges.

The next phase is to add enough edges among points in different X_k's so that no subset of X (other than the $X_{k'}$'s) is a complete partition of Y. We will accomplish this by adding a bunch of points to Y shared by various $x \in X$, so that each x has edges to k_{max} points in Y. These are called the *dispersion* edges.

The final phase is merely to add edges so that all points reach their assigned capacity. We do this arbitrarily. These are called the *filler* edges.

Note whenever we add a point to Y, for all $k \in K_{good}$, we need to add an edge to exactly one $x \in X_k$, which will ensure that all X_k's form a partition of Y.

Phase 1: Backbone edges. Recall that for $k, k' \in K_{good}$, we want $\forall i, \exists j$ such that $x_i^{(k)}$ overlaps with $x_j^{(k')}$. Since $k_{max} = \frac{6}{5} k_{min}$, some x's will be forced to overlap with two points from the same X_k. However, we can ensure no point overlaps with three points from the same X_k.

We satisfy all overlappings naturally by creating k_{min} components, CC_1 to $CC_{k_{min}}$. Each component CC_i contains point $x_i^{(k_{min})}$. The rest of the sets X_k are divided so that one or two points are in each component, as shown in Figure 4.6. Formally, in component CC_i, sets $X_{k_{min}}$ to $X_{k_{min}+\lceil \frac{i}{2} \rceil}$ have one point in the component, and all other sets have two points in the component.

For each component CC_i, we add $\frac{4}{5} n\ell$ points to Y, split into two groups of $\frac{2}{5} n\ell$. The points from sets $X_{k_{min}+\lceil \frac{i}{2} \rceil}$ have edges to all $\frac{4}{5} n\ell$ points, and the points from the rest of the sets (since there are two from each set) have edges to one group of $\frac{2}{5} n\ell$ points. Therefore, for all $k, k' \in K_{good}$, each point $x \in X_k$ belongs to some component CC_i, and overlaps with some $x' \in X_{k'}$, so all of the overlapping requirements are satisfied (only using points within the same component).

This completes phase 1. Each point in X had at most $\frac{4}{5} n\ell$ edges added, so every point can still take at least $\frac{n\ell}{5}$ more edges in subsequent phases.

Phase 2: Dispersion edges. Now we want to add points to Y to ensure that no set of at most k_{max} points from X create a partition of Y, except sets that completely contain some X_k.

We have a simple way of achieving this. For every $(x_1, x_2, \ldots, x_{m+1}) \in X_{k_{min}} \times X_{k_{min}+2} \times \cdots \times X_{k_{max}}$, add one point to Y with edges to $x_1, x_2, \ldots, x_{m+1}$. Then we have added $\prod_{k \in K_{good}} k$ total points to Y in this phase.

This completes phase 2.

Phase 3: Filler edges. The final step is just to fill in the leftover points, since we want every point $x_i^{(k)}$ to have $\frac{k_{min}}{k} l$ points total. All of the mechanisms for the proof have been set up in phases 1 and 2, so these final points can be arbitrary.

We greedily assign points. Give each point $x_i^{(k)} \in X$ a number $t_{x_i^{(k)}} = \frac{k_{min}}{k} n\ell - N(x_i^{(k)})$, i.e., the number of extra points it needs. Take the point $x \in X_k$ with the minimum t, and create t points in Y with x. For each layer other than X_k, add edges to the point with the smallest number. Continue this process until $t = 0$ for all points.

Final Proof. Now we are ready to prove that G has m local maxima. By construction, for all $k \in K_{good}$, X_k is a set of centers which satisfy the capacity constraints, and every point has an edge to its center. Now, consider a set C of centers of size $k' \in K_{bad}$. We show in every case, C cannot satisfy the capacity constraints with all points having edges to their centers.

Case 1: C contains a point $y \in Y$. y only has m edges, which is much smaller than $n\ell$.

Case 2: There exists $k \in K_{good}$ such that $X_k \subseteq C$. Then since $|C| \notin K_{good}$, $\exists x \in C \setminus X_k$. By construction, there exists $x_i^{(k)} \in X_k$ such that x and $x_i^{(k)}$ are overlapping. Therefore, both centers cannot satisfy the capacity constraints with points they have an edge to.

Case 3: For all $k \in K_{good}$, there exists $x \in X_k$ such that $x \notin C$. Take the set of all of these points, $x_1, x_2, \ldots, x_{m+1}$. By construction, there is a point $y \in Y$ with edges to only these points. Therefore, y will not have an edge to its center in this case.

This completes the proof. □

4.5 General Robust Distributed Clustering

In this section, we give a general algorithm for distributed clustering with the ℓ_p objective, with or without balance constraints and with or without outliers. This generalizes previous distributed clustering results [Bateni et al., 2014b, Malkomes et al., 2015], and answers an open question of Malkomes et al. [2015]. 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 (see Figure 4.7). 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 4.5.1. *Given a sequential (δ, α)-approximation algorithm $\mathcal{A}$ [2] 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 Algorithm 18 is 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 [Lin and

[2] We note that $\mathcal{A}$ must be able to handle weighted points. It has been pointed out that every clustering algorithm we are aware of has this property [Bateni et al., 2014b].

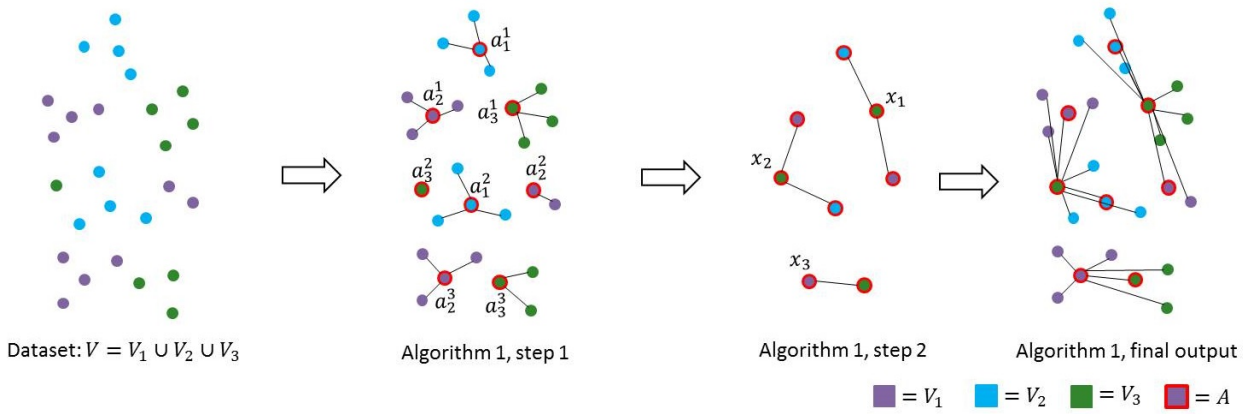


Figure 4.7: Algorithm 18 for k-median, $m = 3$, $k = 3$.

Algorithm 18 Distributed balanced clustering with outliers

Input: Distributed points $V = V_1 \cup \cdots \cup V_m$, algorithms $\mathcal{A}$ and $\mathcal{B}$

1: For each machine i,

- Run $\mathcal{B}$ for $(k+z)$-clustering on V_i, outputting $A_i = \{a_1^i, \ldots, a_{k+z}^i\}$.
- Set $w_j^i = |\{p \in V_j \mid a_j^i = \text{argmin}_{a \in A_i} d(p, a)\}|$.
- Send A_i and all weights to machine 1.

2: Run $\mathcal{A}$ on $\bigcup_i A_i$ using the corresponding weights, outputting $X = \{x_1, \ldots, x_k\}$.

Output: Centers $X = x_1, \ldots, x_k$

Vitter, 1992], the approximation ratio becomes $6\alpha + 2 + \epsilon$, which improves over the 32α approximation ratio of Bateni et al. [2014b]. If we set $\mathcal{A}$ as the current best k-median algorithm [Byrka et al., 2015c], we achieve a distributed $(18.05 + \epsilon)$-approximation algorithm for k-median. If instead we plug in the sequential approximation algorithm for k-median with z outliers [Chen, 2008], we obtain the first constant-factor approximation algorithm for k-median with outliers, answering an open question from Malkomes et al. [2015]. We can also use the results from Gupta and Tangwongsan [2008] to obtain an $O(1)$-approximation algorithm for $1 < p < \log n$.

Our proof of Theorem 4.5.1 carefully 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. First we bound the sum of the local optimal $(k+z)$-clustering on each machine by the global clustering with outliers in the following lemma (a non-outlier version of this lemma appears in [Bateni et al., 2014b]).

Lemma 4.5.2. *For a partition $V_1, \ldots, V_k$ of V and $1 \leq p < \infty$, $\sum_{i=1}^m \mathcal{OPT}_{k+z}(V_i)^p \leq 2^p \mathcal{OPT}_{k,z}^p$.*

Proof. Given a machine with datapoints $V_i \subseteq V$, we will first show that

$$\mathcal{OPT}(V_i, V_i)^p \leq 2^p \mathcal{OPT}(V_i, V)^p.$$

Let $c_1, \ldots, c_k$ be the optimal centers for $\mathcal{OPT}(V_i, V)$. Given c_j, let c_j' be the closest point in V_i to c_j. Note that there may be one point c' which is the closest point in V_i to two different centers, but

this just means we will end up with $\leq k$ centers total, which is okay. Then we have the following:

$$\begin{aligned}
\mathcal{OPT}(V_i, V_i)^p &\leq \sum_{v \in V_i} d(v, c'_v)^p \\
&\leq 2^{p-1} \sum_{v \in V_i} (d(v, c_v)^p + d(c_v, c'_v)^p) \\
&\leq 2^p \sum_{v \in V_i} d(v, c_v)^p \\
&\leq 2^p \mathcal{OPT}(V_i, V)^p
\end{aligned}$$

The third inequality follows because c'_v was defined as the closest point in V_i to c_v. By choosing $k' = k + z$, we have that $\mathcal{OPT}_{k+z}(V_i, V_i)^p \leq 2^p \mathcal{OPT}_{k+z}(V_i, V)^p$ for all i.

Let the centers in $\mathcal{OPT}_{k,z}$ be $c_1, \ldots, c_k$, and for $v \in V$, let c_v denote the closest of these centers to v. Given the outliers Z from $\mathcal{OPT}_{k,z}$, let $V'_i = V_i \setminus Z$. Then $\mathcal{OPT}_{k+z}(V_i, V)^p \leq \mathcal{OPT}_k(V'_i, V)^p \leq \sum_{v \in V'_i} d(v, c_v)^p$. The second inequality follows because with $k + z$ centers, we can make all points in $V_i \cap Z$ a center and also use the centers in $\mathcal{OPT}_k(V'_i, V)$.

Summing over all i, we arrive at $\sum_i \mathcal{OPT}_{k+z}(V_i, V)^p \leq \mathcal{OPT}^p_{k,z}$, and the lemma follows. □

Now we prove Theorem 4.5.1.

Proof. **(Theorem 4.5.1)** Given a (δ, α)-approximation algorithm $\mathcal{A}$ for balanced clustering in ℓ_p with z outliers and a (γ, β)-approximation algorithm $\mathcal{B}$ for ℓ_p clustering, we show that Algorithm 18 outputs a set X of centers with provable approximation guarantees. First we consider the case where $p < \infty$.

We start by defining all the notation we need for the proof. Let Z denote the set of outliers returned by Algorithm 18 when running $\mathcal{B}$, let Z^* denote the outliers in $\mathcal{OPT}_{k,z}(A, A)$, where $A = \bigcup_t A_t$ (defined in Algorithm 18), and let Z' denote the outliers in $\mathcal{OPT}_{k,z}$. Denote the centers in $\mathcal{OPT}_{k,z}(A, A)$ by x^*, denote the centers in $\mathcal{OPT}_{k,z}$ by c_j, and let c'_j denote the closest point in A to c_j. Given a point $v \in V_t$, let a_v,, x_v, and c_v denote the closest point to v in A_t, X, or $\mathcal{OPT}(V)$, respectively. Finally, let c'_v denote the closest point to c_v in A.

Using the triangle inequality and the fact that for all v, $d(v, x_v) \leq d(v, x_{a_v})$,

$$\begin{aligned}
\sum_{v \in V \setminus Z} d(v, x_v)^p \leq \sum_{v \in V \setminus Z} d(v, x_{a_v})^p &\leq 2^{p-1} \sum_{v \in V \setminus Z} (d(v, a_v)^p + d(a_v, x_{a_v})^p) && (4.4) \\
&\leq 2^{p-1} \sum_{v \in V} d(v, a_v)^p + 2^{p-1} \sum_{v \in V \setminus Z} d(a_v, x_{a_v})^p && (4.5)
\end{aligned}$$

We can bound the first summation in Expression 4.5 as follows.

$$\sum_{v \in V} d(v, a_v)^p = \sum_i \sum_{v \in V_i} d(v, a_v)^p \tag{4.6}$$

$$\leq \sum_i \beta^p \mathcal{OPT}_{k+z}(V_i, V_i)^p \qquad \text{[by definition of } \mathcal{B}\text{]} \tag{4.7}$$

$$\leq 2^p \beta^p \mathcal{OPT}^p_{k,z} \qquad \text{[by Lemma 4.5.2.]} \tag{4.8}$$

Now we show how to bound the second summation.

$$\begin{aligned}
\sum_{v \in V \setminus Z} d(a_v, x_{a_v})^p &\leq \alpha^p \sum_{v \in V \setminus Z^*} d(a_v, x_v^*)^p && \text{[by def'n of } \mathcal{A}\text{]} \\
&\leq \alpha^p \sum_{v \in V \setminus Z'} d(a_v, c_v')^p && \text{[by def'n of } \mathcal{OPT}_{k,z}\text{]} \\
&\leq 2^{p-1} \alpha^p \sum_{v \in V \setminus Z'} (d(a_v, c_v)^p + d(c_v, c_v')^p) && \text{[by triangle ineq.]} \\
&\leq 2^p \alpha^p \sum_{v \in V \setminus Z'} d(a_v, c_v)^p && \text{[by definition of } c_v'\text{]} \\
&\leq 2^{2p-1} \alpha^p \sum_{v \in V \setminus Z'} (d(v, a_v)^p + d(v, c_v)^p) && \text{[by triangle ineq.]} \\
&\leq 2^{2p-1} \alpha^p \sum_{v \in V \setminus Z'} d(v, a_v)^p + 2^{2p-1} \alpha^p \sum_{v \in V \setminus Z'} d(v, c_v)^p && \text{[expanding]} \\
&\leq 2^{3p-1} \alpha^p \beta^p \mathcal{OPT}^p_{k,z} + 2^{2p-1} \alpha^p \mathcal{OPT}^p_{k,z} && \text{[by Expression 4.8]} \\
&\leq (2^{3p-1} \alpha^p \beta^p + 2^{2p-1} \alpha^p) \mathcal{OPT}^p_{k,z} && \text{[arithmetic]}
\end{aligned}$$

Therefore, our final result is

$$\left(\sum_{v \in V \setminus Z} d(v, x_v)^p \right)^{\frac{1}{p}} \leq \left(2^{3p-1} \alpha^p \beta^p + 2^{2p-1}(\alpha^p + \beta^p)\right)^{\frac{1}{p}} \mathcal{OPT}_{k,z}.$$

For the communication complexity, it is clear that we communicate $\leq (k+z)m\gamma$ total points and weights, and the weights are numbers $\leq n$, so the communication cost is $O(m(k+z)(d + \log n)\gamma)$. The balance constraints follow from the guarantees of Algorithm $\mathcal{A}$, and the fact that the points in A are weighted. We also note that for all i, machine 1 can send the center assignments of A_i to machine i, so that each datapoint knows its global center (and this does not increase the communication cost above $O(m(k+z)(d+\log n)\gamma)$).

For k-center, we can derive the same result as k-median by using the same analysis, but replacing every summation with a maximum. For instance, we use a modified Lemma 4.5.2 to show $\max_{i=1}^m \mathcal{OPT}_{k+z}(V_i) \leq 2\mathcal{OPT}_{k,z}$. Our final result for k-center is $\max_{v \in V \setminus Z} d(v, x_v) \leq (4\alpha\beta + 2\alpha + 2\beta)\mathcal{OPT}_{k,z}$. □

Chapter 5

Conclusions

In this thesis, we develop the theory of beyond worst-case analysis in multiple areas including perturbation resilience for clustering, data-driven clustering via algorithm configuration, and data-driven dispatching for distributed machine learning. Specifically, we made the following contributions to BWCA.

k-center clustering under perturbation resilience We design robust, efficient algorithms that output near-optimal clusterings under perturbation resilience. We show that any 2-approximation algorithm for symmetric k-center will return the exact solution under 2-perturbation resilience, and we give an algorithm which returns the exact solution for 2-perturbation resilient instances of asymmetric k-center. We prove our results are tight by showing symmetric k-center under $(2-\delta)$-perturbation resilience is hard unless $NP = RP$.

Our work pushes the understanding of (promise) stability conditions farther in several ways. We are the first to design computationally efficient algorithms to find the optimal clustering under α-perturbation resilience with a constant value of α for a problem that is hard to approximate to any constant factor in the worst case, thereby demonstrating the power of perturbation resilience. Furthermore, we demonstrate the limits of this power by showing the first tight results in this space for perturbation resilience. Our work also shows a surprising relation between symmetric and asymmetric instances, in that they are equivalent under resilience to 2-perturbations, which is in stark contrast to their widely differing tight approximation factors.

Finally, we initiate the study of clustering under local stability. We define a local notion of perturbation resilience, and we give algorithms that simultaneously output all optimal clusters from the perturbation resilient regions of the data, while still ensuring the worst-case approximation guarantee. This allows a user to run one single algorithm even if she is not sure whether her input is fully perturbation resilient, partly perturbation resilient, or not at all perturbation resilient, and obtain good results in all cases.

Data-driven clustering We consider a related area of BWCA, in which the goal is to directly optimize for the expected performance over a specific application, which is modeled as an unknown distribution over problem instances. We define rich, infinite families of clustering algorithms, and then show tight bounds on the pseudo-dimension of these families, which leads to computational- and sample-efficient algorithm configuration. More specifically, we give polynomial-time algo-

rithms which output a parameter which is provably close to the parameter with the best expected performance over the unknown distribution, with high probability.

We consider three families of agglomerative algorithms with dynamic programming, and also a family of algorithms generalizing Lloyd's method. The linkage families include the popular single-linkage and complete-linakge aglorithms, and the Lloyd's family includes the celebrated k-means++ algorithm, as well as the classic farthest-first traversal algorithm. We provide sample efficient and computationally efficient algorithms to learn near optimal parameters over an unknown distribution of clustering instances, by developing techniques to bound the expected number of discontinuities in the cost as a function of the parameter. The learned parameters vary among different types of datasets, and the learned parameters often significantly improve the error compared to existing algorithms such as k-means++ and farthest-first traversal.

Distributed machine learning We propose and analyze a new framework for distributed learning. We consider the distributed machine learning model in which data is dispatched to multiple machines for processing and the machines do not communicate during training time. The simplest dispatching algorithm in this case would be random dispatching. Given that similar points tend to have similar classes, we partition the data so that similar examples go to the same machine. We cast the dispatching step as a clustering problem combined with novel fault tolerance and balance constraints necessary for distributed systems. We show the added constraints make the objective highly nontrivial, yet we provide LP rounding algorithms with provable guarantees. These are the first algorithms with provable guarantees under both upper and lower capacity constraints, and may be of interest beyond distributed learning. Finally, we consider the distributed clustering problem. We construct general and robust algorithms for distributed clustering.

Bibliography

Karen Aardal, Pieter L van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, 2:358–368, 2015. 4.1.2, 4.3.1

Margareta Ackerman, Shai Ben-David, and David Loker. Towards property-based classification of clustering paradigms. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 10–18, 2010. 3.1.2

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. 4.1.1

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. 4.1.1

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2017. 4.1.2

Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In *Integer Programming and Combinatorial Optimization*, pages 52–63. Springer, 2014. 4.1.1, 4.1.2, 4.3.1, 4.3.2, 4.3.2, 4.3.2, 4.3.2, 4.3.2, 4.3.14

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. 1, 2.1, 2.1.1, 2.1.2, 2.1.2, 2.2

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009. 3.1

Kohei Arai and Ali Ridho Barakbah. Hierarchical k-means: an algorithm for centroids initialization for k-means. *Reports of the Faculty of Science and Engineering*, 36(1):25–31, 2007. 3.1, 3.1.2

Aaron Archer. Two o (log* k)-approximation algorithms for the asymmetric k-center problem. In *Integer Programming and Combinatorial Optimization*, pages 1–14. Springer, 2001. 2.1.2

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. 2.1.2

David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153. ACM, 2006. 3.1.2

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007. 3.1.1, 3.1.2

David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *Journal of the ACM (JACM)*, 58(5):19, 2011. 3.1.2

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. 2.1

Hassan Ashtiani and Shai Ben-David. Representation learning for clustering: a statistical framework. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, pages 82–91, 2015. 3.1.2

Pranjal Awasthi and Maria-Florina Balcan. Center based clustering: A foundational perspective. 2014. 4.2

Pranjal Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *Proceedings of the International Workshop on Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX-RANDOM)*, pages 37–49. Springer, 2012. 2.1.2

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. 2.1.2

Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012. 2.1, 2.1.2, 3, 9, 3.1.1, 3.1.2, 3.3

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. 3.1.1, 3.1.2, 3.3

Pranjal Awasthi, Maria-Florina Balcan, and Colin White. General and robust communication-efficient algorithms for distributed clustering. *arXiv preprint arXiv:1703.00830*, 2017. 1

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. 4.1.2

Maria-Florina Balcan and Mark Braverman. Finding low error clusterings. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 3–4, 2009. 2.1.2

Maria-Florina Balcan and Mark Braverman. Nash equilibria in perturbation-stable games. *Theory of Computing*, 13(13):1–31, 2017. 2.1.2

Maria Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016. 2.1, 2.1.2, 2.3, 3, 10, 3.1.1, 3.1.2, 3.3, 3.3

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. 1

Maria Florina Balcan, Heiko Röglin, and Shang-Hua Teng. Agnostic clustering. In *International Conference on Algorithmic Learning Theory*, pages 384–398, 2009. 2.1.2

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. 4.1

Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *Journal of the ACM (JACM)*, 60(2):8, 2013a. 2.1, 2.1.2, 5, 17

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2013b. 4.1

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. 1, 3.1.1, 3.1.2, 3.3

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. 1

Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized lloyd's families. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2018. 1, 3.1.1, 3.4

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, 2013c. 4.1, 4.1.2

Judit Bar-Ilan, Guy Kortsarz, and David Peleg. How to allocate network centers. *Journal of Algorithms*, 15(3):385 – 415, 1993. 4.1.2

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002. 3.4.1

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, 2014a. 1

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, 2014b. 4.1, 4.1.1, 4.1.2, 4.2, 4.5, 2, 4.5

Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39, 1997. 3.1, 3.1.2

Shalev Ben-David and Lev Reyzin. Data stability in clustering: A closer look. In *Algorithmic Learning Theory*, pages 184–198. Springer, 2012. 2.3.3, 2.3.3

Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(05):643–660, 2012. 1, 2.1, 2.1.2, 2.2

Florian Bourse, Marc Lelarge, and Milan Vojnovic. Balanced graph edge partition. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1456–1465, 2014. 4.1.2

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. 4.1.2, 4.2

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, 2015a. 4.1.2

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, 2015b. 4.1.1

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, 2015c. 2.1, 4.1.2, 4.5

Fazli Can. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems (TOIS)*, 11(2):143–164, 1993. 3.1.1

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. 3.1, 3.1.2

Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 626–635, 1997. 3.1.1

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, 1999a. 4.1.2, 4.3.1, 4.3.1

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, 1999b. 2.1, 4.1.2

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. 2.1, 4.1.2

Chandra Chekuri and Shalmoli Gupta. Perturbation resilient clustering for k-center and related problems via lp relaxations. In *Proceedings of the International Workshop on Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX-RANDOM)*, 2018. 2.1.2

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. 4.1.2

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. 2.1, 4.1.1, 4.1.2, 4.5

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. 2.1.1, 2.1.2, 2.5.2

Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 9–21. ACM, 2016. 3.1

Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2017. 2.1.2

Brian F Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, 1(2):1277–1288, 2008. 4.1.2

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. 4.1.1, 4.1.2, 4.3.1, 4.3.2, 4.3.2, 4.3.2, 4.3.9, 4.3.2

Sanjoy Dasgupta and Philip M Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005. 3.1.1, 3.1.2

Daniel Delling, Andrew V Goldberg, Ilya Razenshteyn, and Renato F Werneck. Graph partitioning with natural cuts. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1135–1146, 2011. 4.1.2

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. 3.1, 3.1.2

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society*, pages 1–38, 1977. 3.1.2

Amit Deshpande, Anand Louis, and Apoorv Vikram Singh. Clustering perturbation resilient instances. *arXiv preprint arXiv:1804.10827*, 2018. 2.1.2

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. 1, 4.1.1

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. 3.2

Martin E Dyer and Alan M Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, 1985. 2.1.2, 2.3.2

Martin E. Dyer and Alan M. Frieze. Planar 3dm is np-complete. *Journal of Algorithms*, 7(2): 174–184, 1986. 2.3.3

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer, 2001. 3.1

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. 3.1.2

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. 4.1.2

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. 1, 2.1, 2.1.2, 2.3.2, 2.6.4, 3.1.1, 3.1.2, 3.4, 4.1.2

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. 3.1.1, 3.3

Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008. 1, 4.5

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. 1, 3.1, 3.1.2, 3.2.1

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. 2.1.2

Sariel Har-Peled and Bardia Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005. 3.1.2

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. 2.1, 2.1.2

Richard E Higgs, Kerry G Bemis, Ian A Watson, and James H Wikel. Experimental designs for selecting molecules from large chemical databases. *Journal of chemical information and computer sciences*, 37(5):861–870, 1997. 3.1, 3.1.2

Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985. 2.1.2, 2.3.2, 2.6.3

Harry B Hunt III, Madhav V Marathe, Venkatesh Radhakrishnan, and Richard E Stearns. The complexity of planar counting problems. *SIAM Journal on Computing*, 27(4):1142–1167, 1998. 2.3.3

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. 1, 2.1, 2.6.4

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. 4.3

Wenhao Jiang and Fu-lai Chung. Transfer spectral clustering. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, pages 789–803, 2012. 3.1.2

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002. 3.1.2

Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009. 3.1.2

Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, pages 152–166, 1996. 4.1.1, 4.1.2, 4.3.1, 4.3.2, 4.3.2

Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006. 2.3.3

Jon M Kleinberg. An impossibility theorem for clustering. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 463–470, 2003. 3.1.2

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. An online hierarchical algorithm for extreme clustering. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, 2017. 3.4

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. 2.1.2

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. 2.1, 2.1.2

Hunter Lang, David Sontag, and Aravindan Vijayaraghavan. α-expansion is exact on stable instances. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 1050, page 6, 2017. 2.1.2

Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017. 1, 2.1, 2.6.4

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. 3.1, 3.1.2

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. 4.1

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. 4.1.1, 4.1.2, 4.3, 4.3.1, 4.3.1, 4.3.1, 4.3.1

Shi Li. Approximating capacitated k-median with $(1 + \epsilon)k$ open facilities. *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, 2016. 4.1.1

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. 4.1

Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992. 4.1.1, 4.5

Stuart Lloyd. Least squares quantization in pcm. *transactions on information theory*, 28(2):129–137, 1982. 3.1, 3.1.2

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 3.1.2

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. 2.1, 2.1.2

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. 2.1, 4.1.2

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. 4.1, 4.1.1, 4.1.2, 4.2, 4.5, 4.5

Bodo Manthey and Matthijs B Tijink. Perturbation resilience for the facility location problem. *Operations research letters*, 46(2):215–218, 2018. 2.1.2

Pascal Massart. Some applications of concentration inequalities to statistics. In *Annales-Faculte des Sciences Toulouse Mathematiques*, volume 9, pages 245–303. Université Paul Sabatier, 2000. 3.4.1

Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1): 7–12, 1960. 3.1.2

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. 2.1.2

Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):28, 2012. 2.1, 2.1.2, 3.1.2

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998. 4.3.1

Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, pages 277–281, 1999. 3.1.2

José M Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999. 3.1, 3.1.2

David Pollard. *Convergence of stochastic processes*. Springer-Verlag, 1984. 3.1

David Pollard. *Empirical processes*. Institute of Mathematical Statistics, 1990. 3.1

Kim D Pruitt, Tatiana Tatusova, Garth R Brown, and Donna R Maglott. Ncbi reference sequences (refseq): current status, new features and genome annotation policy. *Nucleic acids research*, 40 (D1):D130–D135, 2011. 3.4

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 759–766, 2007. 3.1.2

John R Rice. The algorithm selection problem. *Advances in computers*, 15:65–118, 1976. 3.1, 3.1.2

Tim Roughgarden. Beyond worst-case analysis. http://theory.stanford.edu/ tim/f14/f14.html, 2014. 2.1, 2.1.2

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. 3.1.1, 3.1.2, 3.3

Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2110–2118, 2016. 3.1.2

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. 1, 2.1

Timo Tossavainen. On the zeros of finite sums of exponential functions. *Australian Mathematical Society Gazette*, 33(1):47–50, 2006. 3.3.6

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 3.1.2

Leslie G Valiant and Vijay V Vazirani. Np is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986. 2.3.3, 2.3.3

Vladimir N. Vapnik and Leon Bottou. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 1993. 1, 4.1

Andrea Vattani. K-means requires exponentially many iterations even in the plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011. 3.1.2

Aravindan Vijayaraghavan, Abhratanu Dutta, and Alex Wang. Clustering stable instances of euclidean k-means. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 6503–6512, 2017. 2.1.2

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. 2.1.1, 2.1.2, 2, 2.5, 2.5.2, 2.5.2, 2.5.2, 2.5.2, 2, 2.5.2, 2.6.1

Konstantin Voevodski, Maria-Florina Balcan, Heiko Röglin, Shang-Hua Teng, and Yu Xia. Min-sum clustering of protein sequences with limited distance information. In *International Workshop on Similarity-Based Pattern Recognition*, pages 192–206, 2011. 2.1.2

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. 4.1.2

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. 3.1.1, 3.1.2, 3.3

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. 3.1, 3.1.2

Qiang Yang, Yuqiang Chen, Gui-Rong Xue, Wenyuan Dai, and Yong Yu. Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Conference on Natural Language Processing*, pages 1–9, 2009. 3.1.2

Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. CA-SVM: Communication-avoiding support vector machines on clusters. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2015. 4.1.2

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. 4.1, 4.1.2

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. 4.1, 4.1.2