MODERN ASPECTS OF UNSUPERVISED LEARNING

A Thesis
Presented to
The Academic Faculty

by

Yingyu Liang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
July 2014

MODERN ASPECTS OF UNSUPERVISED LEARNING

Approved by:

Maria-Florina Balcan, Advisor
School of Computer Science
Georgia Institute of Technology

Avrim Blum
School of Computer Science
Carnegie Mellon University

Lance Fortnow
School of Computer Science
Georgia Institute of Technology

Charles L. Isbell, Jr.
School of Interactive Computing
Georgia Institute of Technology

Dana Randall
School of Computer Science
Georgia Institute of Technology

Le Song
School of Computational Science and Engineering
Georgia Institute of Technology

Date Approved: 1 June 2014

To my parents,

Qunhong Liang and Guixian Qin.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Maria-Florina Balcan, for her excellent guidance, caring, and patience. She has been a tremendous mentor for me, providing me with an excellent model for doing research with great enthusiasm and sharp insights. Her advice on both research as well as on my career has been priceless. I would also like to thank my committee members, Avrim Blum, Lance Fortnow, Charles L. Isbell, Jr., Dana Randall, and Le Song for serving as my committee members and providing stimulating suggestions and comments for this thesis.

I would like to thank Lev Reyzin, Steven Ehrlich, and Christopher Berlind, who were always willing to help and give their best suggestions on my research and my writing. Many thanks to Anand Louis, Ying Xiao, Arindam Khan, Abhishek Banerjee, Shang-Tse Chen, Nan Du, Bo Dai, He Niao and many others in the laboratory who have made my years at Georgia Institute of Technology wonderful and productive.

I would like to express my special thanks to my family. Words cannot express how grateful I am to my parents, parents-in-law, my sisters and brother for all of the sacrifices that they have made on my behalf. They are always supporting me and encouraging me with their best wishes. At the end, I would like to thank my beloved wife Ting Jiao, who is always there cheering me up and stands by me through all times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Unsupervised learning has become more and more important due to the recent explosion of data. Clustering, a key topic in unsupervised learning, is a well-studied task arising in many applications ranging from computer vision to computational biology to the social sciences. This thesis is a collection of work exploring two modern aspects of clustering: stability and scalability.

In the first part, we study clustering under a stability property called perturbation resilience. As an alternative approach to worst case analysis, this novel theoretical framework aims at understanding the complexity of clustering instances that satisfy natural stability assumptions. In particular, we show how to correctly cluster instances whose optimal solutions are resilient to small multiplicative perturbations on the distances between data points, significantly improving existing guarantees. We further propose a generalized property that allows small changes in the optimal solutions after perturbations, and provide the first known positive results in this more challenging setting.

In the second part, we study the problem of clustering large scale data distributed across nodes which communicate over the edges of a connected graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. We also propose algorithms for distributed principal component analysis, which can be used to reduce the communication cost of clustering high dimensional data while merely comprising the clustering quality.

In the third part, we study community detection, the modern extension of clustering to network data. We propose a theoretical model of communities that are

stable in the presence of noisy nodes in the network, and design an algorithm that provably detects all such communities. We also provide a local algorithm for large scale networks, whose running time depends on the sizes of the output communities but not that of the entire network.

CHAPTER I

INTRODUCTION

This thesis develops new frameworks and designs algorithms for new aspects of clustering, a key topic in unsupervised learning. These modern aspects are areas of significant practical importance and rising concerns. Traditional clustering analysis tends to not capture these new aspects, hence new frameworks and algorithms are desirable.

Generally, the goal of clustering is to identify meaningful subsets (called clusters) within a set of data points. The clusters are selected in such a way so that the points in the same clusters are more similar to each other than to the points in different clusters. Clustering is unsupervised because it does not make use of annotated data in order to estimate the aforementioned clusters. Instead, the clusters are identified only by using the characteristics of the data points, such as the feature vector representation of the points, or the distances between them.

There are different paradigms for clustering in the literature, among which are the following common approaches.

- Objective-based clustering, which impose a quantitative objective and assume that the target clustering is equal or close to the partition that optimizes the objective. For example, in the classic k-means clustering [81, 5], the points are partitioned into k clusters and each cluster is assigned a center, and the objective is the sum of the squared distances between the points and their centers. Another example is k-median clustering, whose objective is the sum of the distances between the points and their centers.

- Hierarchical clustering, which builds a hierarchy (typically a tree) of clusters

instead of a partition. This includes both agglomerative and divisive algorithms. The agglomerative algorithms typically begin with each point being a cluster, and then repeatedly merge the two closest clusters, where the closeness between clusters is specified by some criterion such as single-linkage, complete-linkage, or average-linkage [98, 66, 44]. The divisive algorithms typically begin with all points in one cluster, and then repeatedly split one current cluster into several clusters based on some subroutine, such as k-means [100] or spectral clustering [31].

- Distribution-based clustering, which models the generative distributions of the data by using statistics and probabilities. For example, the mixture of Gaussians model [37] assumes the points are generated from a mixture of Gaussian distributions, and aims at recovering the parameters of the mixture.

- Graph clustering, which organizes the data on the basis of the edge structure between the points and identifies the clusters by studying the edge structure. The edges can be deterministically given by the application, such as the links between the members of a social network. The clusters can then be identified by optimizing some criterion, such as conductance [61], or cut ratio [85]. In this case, the approach is closely related to the objective-based clustering. The edges can also be generated from some distribution. For example, in the planted k-partition model [34], the points are divided into k clusters, and points in the same clusters are connected with probability p while points in different clusters are connected with probability q, where p, q are two parameters. The clusters can then be identified by estimating the distribution parameters. In this case, the approach is closely related to the distribution-based clustering. In some literature on network analysis, graph clustering is also called community detection [51, 90].

- Set system clustering, which specifies some properties the collection of clusters should satisfy and aims at finding such a collection. For example, [22] aimed at finding the collection of clusters such that for any two points p and q in the same cluster and any point w outside the cluster, w is farther away from at least one of p and q than p is from q. Some other work [67, 1] specifies a set of axioms for the collection of clusters, and studies the existence and uniqueness of such a collection.

This thesis considers the objective-based clustering where we are given the number k of clusters and the distances between the data points. First, it is one of the most frequently used and studied approaches. Second, as described in the paragraphs above, the other approaches are closely related to the objective-based clustering. For example, divisive hierarchical clustering algorithms may have a subroutine that is objective-based, while estimating the parameters in distribution-based clustering may reduce to some objective function over the sampled points. Third, although it has been well studied, there are new challenges that are not (fully) addressed by existing analysis. This thesis also considers graph clustering (named community detection when applied to network analysis) due to the recent explosion of network data and the increasing interest in understanding them.

We focus on two modern aspects of the two clustering approaches considered: stability and scalability.

First, since optimizing most natural clustering objectives is **NP**-hard, there has been an increasing interest in beyond worst-case analysis of clustering based on the stability properties of the target [92, 14, 9, 26, 69, 8]. Such stability properties formalize what is implicitly assumed about the practically interesting instances. Bilu and Linial [26] proposed a property called perturbation resilience, which assumes that the optimal solution does not change under small multiplicative perturbation to the pairwise distances. The first part of this thesis provides a study of this property and

its generalization for center-based clustering (including k-median and k-means) and min-sum clustering.

Another trend in the clustering task is to deal with large scale data, especially those collected in the distributed setting. A natural question in this setting is how to compute high quality solution with low communication. Due to the constraint on communication, most distributed applications only ask for a good approximation solution. Ideally, there is a tradeoff between the communication and the solution quality, so that the application can choose its sweet spot between the two: with more communication, the approximation guarantee gets better; any given small error can be achieved with a suitable amount of communication. The second part of the thesis provides a study of such a tradeoff, and further discusses how to reduce the communication of clustering high dimensional data by distributed principal component analysis.

Finally, the third part of the thesis turns to community detection on network data. We propose a model of communities based on some stability property with respect to noisy nodes in the network, and design an algorithm which provably detects all such stable communities. Furthermore, we design a local algorithm to handle large scale networks, whose running time of finding a community for any given node is independent of the size of the entire network.

1.1 Clustering under Perturbation Resilience

For most natural clustering objectives, finding the optimal solution to the objective function is **NP**-hard. As a consequence, there has been substantial work on approximation algorithms [59, 29, 23, 40, 6] with both upper and lower bounds on the approximability of these objective functions on worst case instances. Bilu and Linial [26] suggested an interesting alternative approach aimed at understanding the complexity of clustering instances which arise in practice. They argued that interesting instances

should be resilient to small perturbations in the distances, and specifically defined an instance to be α-perturbation resilient for an objective Φ if perturbing pairwise distances by multiplicative factors in the range $[1, \alpha]$ does not change the optimum clustering under Φ. Two important questions raised are: (1) how much resilience is required so that one can develop algorithms for important clustering objectives? (2) the resilience definition requires the optimum solution to remain *exactly* the same after perturbation: can one succeed under weaker conditions?

In Chapter 2, we address both these questions. First, for the center-based objectives, we design a polynomial time algorithm for finding the optimum solution for instances resilient to perturbations of value $\alpha = 1 + \sqrt{2}$, thus beating the previously best known factor of 3 of Awasthi et al [9]. Second, for k-median (which is a specific center-based objective), we consider a weaker, relaxed, and more realistic notion of perturbation-resilience where we allow the optimal clustering of the perturbed instance to differ from the optimal of the original in a small ϵ fraction of the points. Compared to the original perturbation resilience assumption, this is arguably a more natural though also more difficult condition to deal with. We give positive results for this case as well, showing for $\alpha > 4$ we can in polynomial time compute a $(1 + O(\epsilon/\rho))$-approximation to the optimum, where ρ is the fraction of the points in the smallest cluster.

We further give positive results for min-sum clustering, which is a generally much harder objective than center-based objectives. For α-perturbation resilient min-sum instances, we provide the first efficient algorithm for optimally clustering when α is in the order of the ratio between the sizes of the largest and smallest clusters. For (α, ϵ)-perturbation resilient min-sum instances with α in the order of the ratio between the sizes of the largest and smallest clusters and $\epsilon = \tilde{O}(\rho)$, we provide a polynomial time algorithm that outputs a clustering that is both a $(1 + \tilde{O}(\epsilon/\rho))$-approximation and $\tilde{O}(\epsilon)$-close to the optimal clustering.

We additionally provide sublinear-time algorithms both for the k-median and min-sum objectives, showing algorithms that can return an implicit clustering from only access to a small random sample.

1.2 Distributed Clustering

Most classic clustering algorithms are designed for the centralized setting, but in recent years large scale data has become distributed over different locations. As a consequence, it has become crucial to develop clustering algorithms which are effective in the distributed setting. Several algorithms for distributed clustering have been proposed and empirically tested. Some of these algorithms [48, 101, 38] are direct adaptations of centralized algorithms which rely on statistics that are easy to compute in a distributed manner. Other algorithms [60, 64] generate summaries of local data and transmit them to a central coordinator which then performs the clustering algorithm. No theoretical guarantees are provided for the clustering quality in these algorithms, and they do not try to minimize the communication cost. Additionally, most of these algorithms assume that the distributed nodes can communicate with all other sites or that there is a central coordinator that communicates with all other sites.

In Chapter 3, we study the problem of distributed clustering where the data is distributed across nodes which communicate over the edges of an arbitrary connected graph. We provide algorithms for k-means and k-median clustering, which have low communication costs and provable guarantees on the clustering quality. Our technique for reducing communication in general graphs is based on the construction of a small set of points called coreset [57], which act as a proxy for the entire data set. An ϵ-coreset is a weighted set of points whose cost on any set of centers is approximately the cost of the original data on those same centers (up to a multiplicative factor $1+\epsilon$), thus an α-approximate solution for the coreset is also an $\alpha(1+O(\epsilon))$-approximate solution

for the original data. In our distributed algorithms for k-means and k-median, each node constructs a local portion of a global coreset. Communicating the total cost of local approximate solutions to each node is enough for the construction, leading to low communication cost overall. The nodes then share the local portions of the coreset, which can be done efficiently in general graphs using a message passing approach. The total communication cost then is proportional to the size of the coreset, which is $\tilde{O}(kd + sk)$ where d is the dimension of the data point and s is the number of the nodes in the communication graph.

The size of the coreset is independent of the size of the original data, which is useful for large-scale applications. However, it is linear in the dimension of the data, leading to a communication cost quadratic in the dimension, which is not scalable to high dimensional data. We propose a distributed principal component analysis algorithm, which projects the data to a subspace of dimension $O(k/\epsilon^2)$ with a communication cost of $O(skd/\epsilon^2)$ words. We show that its output represents the original data in the sense that any α-approximation solution of k-means clustering on the output projected data is an $\alpha(1 + \epsilon)$-approximation solution on the original data. We can then apply the aforementioned coreset based distributed clustering algorithm, whose communication cost is now independent of the size and dimension of the original data.

1.3 Community Detection

In analyzing social network, it is meaningful to identify interesting subsets called communities based on the affinities between the members (nodes) of the social network. In Chapter 4, we propose a theoretical model that formalizes the collection of target communities. In our model, each node of a community falls into a sub-community and the sub-communities within this community have active interactions with each other, while entities outside this community have fewer interactions with nodes inside. To deal with practical noise, we only require the communities to satisfy the

above property after removing a few nodes from the network. This means that such communities are stable in the presence of a few exceptional or irregular outliers (such as a member connected to almost all other members, or a member whose affinity data with others have been corrupted). Given this formalization, we then propose an efficient algorithm that detects all the communities in this model, and prove that all the communities form a tree hierarchy. Furthermore, a local algorithm is designed to handle large scale networks. Such an algorithm takes a node in the network as input, and outputs a community containing this node. Its running time depends on the size of the output community but not that of the entire network, and thus it is particularly suitable for large scale networks. Empirical evaluations demonstrate that our formalization successfully models real world communities, and our algorithm compares favorably with existing approaches.

1.4 Summary and Bibliographic Information

The thesis is organized as follows.

- In Chapter 2 we study the objective-based clustering under a stability notion called perturbation resilience. We improve the existing bound for center-based objectives including k-median and k-means, and provide the first known bound for the min-sum objective. We further propose a generalization of perturbation resilience, and provide the first known bounds under this generalization for the k-median and min-sum objectives. Finally, we provide sublinear algorithms for the k-median and min-sum objectives. Part of this chapter is based on the work that appears in [19].

- In Chapter 3 we provide algorithms for k-means and k-median clustering in the distributed setting where the data is distributed across nodes which communicate over the edges of a connected graph, and bound the communication cost

and clustering quality in our algorithms. For k-means clustering of high dimensional data, we further provide an algorithm that first reduces the dimension by distributed PCA and then performs clustering. This chapter is mostly based on the work that appears in [78] and [79].

- In Chapter 4 we propose a model for communities over networks, and provide an algorithm that provably detects all the communities. We further propose a local algorithm for large scale networks. Part of this chapter is based on the work that appears in [20].

Other work that we have done but not included in the thesis includes the following. In [17], we study learning disjunctions in the semi-supervised and active setting, and provide efficient algorithms with nearly optimal label complexity. In [24], we study the problem of learning sparse combinations of elements distributed across a network, and propose a distributed Frank-Wolfe algorithm that solves this class of problems in a scalable and communication-efficient way. In [43], we provide an efficient algorithm for learning the spread of information over networks, based on the insight that the influence functions in many diffusion models are coverage functions and can be learned by convex combinations of random basis functions. In [42], we study maximizing the influence of multiple information items based on the learned influence function, and provide a novel formulation as a submodular maximization under the intersection of a matroid and multiple knapsack constraints, the special structure of which leads to an efficient algorithm with an approximation factor better than known guarantees.

CHAPTER II

CLUSTERING UNDER PERTURBATION RESILIENCE

As discussed in the introduction, a common approach for solving clustering problems is objective-based clustering. We are given the number k of clusters and the distances between the data points, and we want to optimize various objective functions such as k-median or min-sum. For example, in the k-median clustering problem the goal is to partition the data into k clusters C_i, giving each a center c_i, in order to minimize the sum of the distances of all data points to the centers of their cluster. In the min-sum clustering approach, the goal is to find k clusters C_i that minimize the sum of all intra-cluster pairwise distances. Yet unfortunately, for most natural clustering objectives, finding the optimal solution to the objective function is **NP**-hard. As a consequence, there has been substantial work on approximation algorithms [59, 29, 23, 40, 6] with both upper and lower bounds on the approximability of these objective functions on worst case instances.

Recently, Bilu and Linial [26] suggested an exciting, alternative approach aimed at understanding the complexity of clustering instances which arise in practice. Motivated by the fact that distances between data points in clustering instances are often based on a heuristic measure, they argue that interesting instances should be resilient to small perturbations in these distances. In particular, if small perturbations can cause the optimum clustering for a given objective to change drastically, then that probably is not a meaningful objective to be optimizing. Bilu and Linial [26] specifically define an instance to be α-perturbation resilient[1] for an objective Φ if perturbing pairwise distances by multiplicative factors in the range $[1, \alpha]$ does not change the

[1]Bilu and Linial [26] refer to such instances as perturbation stable instances.

optimum clustering under Φ.[2] They consider in detail the case of Max-Cut clustering and give an efficient algorithm to recover the optimum when the instance is resilient to perturbations on the order of $\alpha = O(\sqrt{n})$.

Two important questions raised by the work of Bilu and Linial [26] are: (1) the degree of resilience needed for their algorithm to succeed is quite high: can one develop algorithms for important clustering objectives that require much less resilience? (2) the resilience definition requires the optimum solution to remain *exactly* the same after perturbation: can one succeed under weaker conditions? In the context of *center-based* clustering objectives such as k-median and k-center, [9] partially address the first of these questions and show that an algorithm based on the single-linkage heuristic can be used to find the optimal clustering for α-perturbation-resilient instances for $\alpha = 3$. They also conjecture it to be **NP**-hard to beat 3 and prove beating 3 is **NP**-hard for a related notion.

In this work, we address both questions raised by [26] and additionally improve over [9]. First, for the center-based objectives we design a polynomial time algorithm for finding the optimum solution for instances resilient to perturbations of value $\alpha = 1+\sqrt{2}$, thus beating the previously best known factor of 3 in [9]. Second, for k-median (which is a specific center-based objective), we consider a weaker, relaxed, and more realistic notion of perturbation-resilience where we allow the optimal clustering of the perturbed instance to differ from the optimal of the original in a small ϵ fraction of the points. Compared to the original perturbation resilience assumption, this is arguably a more natural though also more difficult condition to deal with. We give positive results for this case as well, showing for somewhat larger values of α that we can still achieve a near-optimal clustering on the given instance (see Section 1.1 below for precise results). We additionally give positive results for min-sum clustering which is

[2]Of course, the *score* of the optimum solution will change; what the definition requires is that the partitioning induced by the optimum remains the same.

a generally much harder objective than center-based objectives. For example, the best known guarantee for min-sum clustering on worst-case instances is an $O(\delta^{-1}\log^{1+\delta} n)$-approximation algorithm that runs in time $n^{O(1/\delta)}$ due to Bartal et al. [23]; by contrast, the best guarantee known for k-median is factor $1+\sqrt{3}+\epsilon$ [76].

Our results are achieved by carefully deriving structural properties of perturbation-resilience. At a high level, all the algorithms we introduce work by first running appropriate linkage procedures to produce a hierarchical clustering, and then running dynamic programming to retrieve the best k-clustering present in the tree. To ensure that (under perturbation resilient instances) the hierarchy output in the first step has a pruning of low cost, we derive new linkage procedures (closure linkage and robust average linkage) which are of independent interest. While the overall analysis is quite involved, the clustering algorithms we devise are simple and robust. This simplicity and robustness allow us to show how our algorithms can be made sublinear-time by returning an implicit clustering from only a small random sample of the input.

From a learning theory perspective, the resilience parameter, α, can also be seen as an analog to a margin for clustering. In supervised learning, the margin of a data point is the distance, after scaling, between the data point and the decision boundary of its classifier, and many algorithms have stronger guarantees when the smallest margin over the entire data set is sufficiently large [96, 102]. The α parameter, similarly controls the magnitude of the perturbation the data can withstand before being clustered differently, which is, in essence, the data's distance to the decision boundary for the given clustering objective. Hence, perturbation resilience is also a natural and interesting assumption to study from a learning theory perspective.

Our Results: In this work, we greatly advance the line of work of [26] by solving a number of important problems of clustering perturbation-resilient instances under metric center-based and min-sum objectives.

In Section 2.2 we improve on the bounds of [9] for α-perturbation resilient instances

for center-based objectives, giving an algorithm that efficiently[3] finds the optimum clustering for $\alpha = 1 + \sqrt{2}$. Most of the frequently used center-based objectives, such as k-median, are **NP**-hard to even approximate, yet we can recover the exact solution for perturbation resilient instances. Our algorithm is based on a new linkage procedure using a new notion of distance (closure distance) between sets that may be of independent interest.

In Section 2.3 we consider the more challenging and more general notion of (α, ϵ)-perturbation resilience for k-median, where we allow the optimal solution after perturbation to be ϵ-close to the original. We provide an efficient algorithm which for $\alpha > 4$ produces $(1+O(\epsilon/\rho))$-approximation to the optimum, where ρ is the fraction of the points in the smallest cluster. The key structural property we derive and exploit is that, except for ϵn bad points, most points are α times closer to their own center than to any other center. To eliminate the noise introduced by the bad points, we carefully partition the points into a list of sufficiently large blobs, each of which contains only good points from one optimal cluster. This then allows us to construct a tree on the blobs with a low-cost pruning that is a good approximation to the optimum.

In Section 2.4 we provide the first efficient algorithm for optimally clustering α-perturbation resilient min-sum instances. Our algorithm is based on an appropriate modification of average linkage that exploits the structure of min-sum perturbation resilient instances.

In Section 2.5, we show that for (α, ϵ)-perturbation resilient min-sum instances with α in the order of the ratio between the sizes of the largest and smallest clusters and $\epsilon = \tilde{O}(\rho)$, there exists a polynomial time algorithm that outputs a clustering that is both a $(1 + \tilde{O}(\epsilon/\rho))$-approximation and $\tilde{O}(\epsilon)$-close to the optimal clustering.

We also provide sublinear-time algorithms both for the k-median and min-sum

[3]For clarity, in this paper efficient means polynomial in both n (the number of points) and k (the number of clusters).

objectives (Sections 2.3.3 and 2.4), showing algorithms that can return an implicit clustering from only access to a small random sample.

Related Work: A subsequent work of [26] by Bilu, Daniely, Linial and Saks [27] studied the Max-Cut problem under Bilu and Linial stability, and showed how to solve in polynomial time $(1+\epsilon)$-stable instances of metric and dense Max-Cut, and $\Omega(\sqrt{n})$-stable instances of general Max-Cut. The later bound is further improved by Makarychev, Makarychev and Vijayaraghavan [84]. They proposed a polynomial time exact algorithm for $\Omega(\sqrt{\log n}\log\log n)$-stable Max-Cut instances based on semidefinite programming. They also proved that for Max k-Cut with $k \geq 3$, there is no polynomial-time algorithm that solves ∞-stable instances of Max k-Cut unless $\mathbf{NP} = \mathbf{RP}$. Here an instance is ∞-stable if it is α-stable for every α.

In the context of objective based clustering, several recent papers have shown how to exploit other notions of stability for overcoming the existing hardness results on worst case instances. The ORSS stability notion of Ostrovsky, Rabani, Schulman and Swamy [92, 9] assumes that the cost of the optimal k-means solution is small compared to the cost of the optimal $(k-1)$-means solution. The BBG approximation stability condition of Balcan, Blum and Gupta [14] assumes that every nearly optimal solution is close to the target clustering. Awasthi, Sheffet and Blum [8] proposed a stability condition called weak-deletion stability, and showed that it is implied by both the ORSS stability and the BBG stability. Kumar and Kannan [69] proposed a proximity condition which assumes that in the target clustering, most data points satisfy that they are closer to their center than to any other center by an additive factor in the order of the maximal standard variance of their clusters in any direction. Their results are improved by Awasthi and Sheffet [10], which proposed a weaker version of the proximity condition called center separation, and designed algorithms achieving stronger guarantees under this weaker condition.

Several recent papers have shown how to exploit the structure of perturbation

resilient instances in order to obtain better approximation guarantees (than those possible on worst case instances) for other difficult optimization problems. These include the game theoretic problem of finding Nash equilibria [15, 80] and the classic traveling salesman problem [86].

2.1 Preliminaries

In a clustering instance, we are given a set P of n points in a finite metric space, and we denote $d : P \times P \to \mathbf{R}_{\geq 0}$ as the distance function. Φ denotes the objective function over a partition of P into $k < n$ clusters which we want to optimize over the metric, ie. Φ assigns a score to every clustering. The optimal clustering with respect to Φ is denoted as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, and its cost is denoted as $\mathcal{OPT}$. The core concept we study in this paper is the perturbation resilience notion introduced by [26]. Formally:

Definition 1. *A clustering instance (P, d) is α-**perturbation resilient** to a given objective Φ if for any function $d' : P \times P \to \mathbf{R}_{\geq 0}$ s.t. $\forall p, q \in P, d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$, there is a unique optimal clustering $\mathcal{C}'$ for Φ under d' and this clustering is equal to the optimal clustering $\mathcal{C}$ for Φ under d.*

In this paper, we focus on the center-based and min-sum objectives. For the *center-based objectives*, we consider separable center-based objectives defined by [9].

Definition 2. *A clustering objective is center-based if the optimal solution can be defined by k points $c_1, \cdots, c_k$ in the metric space called centers such that every data point is assigned to its nearest center. Such a clustering objective is separable if it furthermore satisfies the following two conditions:*

- *The objective function value of a given clustering is either a (weighted) sum or the maximum of the individual cluster scores.*
- *Given a proposed single cluster, its score can be computed in polynomial time.*

One particular center-based objective is the k-median objective. We partition P into k disjoint subsets $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ and assign a set of centers $\mathbf{p} = \{p_1, p_2, \ldots, p_k\} \subseteq P$ for the subsets. The goal is to minimize $\Phi(\mathcal{P}, \mathbf{p}) = \sum_{i=1}^{k} \sum_{p \in P_i} d(p, p_i)$. The centers in the optimal clustering are denoted as $\mathbf{c} = \{c_1, \ldots, c_k\}$. Clearly, in an optimal solution, each point is assigned to its nearest center. In such cases, the objective is denoted as $\Phi(\mathbf{c})$.

For the *min-sum objective*, we partition P into k disjoint subsets $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, and the goal is to minimize $\Phi(\mathcal{P}) = \sum_{i=1}^{k} \sum_{p,q \in P_i} d(p, q)$. Note that we sometimes denote Φ as Φ_P in the case where the distinction is necessary, such as in Section 2.3.3.

In Section 2.3 we consider a generalization of Definition 1 where we allow a small difference between the original optimum and the new optimum after perturbation. Formally:

Definition 3. *Let $\mathcal{C}$ be the optimal k-clustering and $\mathcal{C}'$ be another k-clustering of a set of n points. We say $\mathcal{C}'$ is ϵ-**close** to $\mathcal{C}$ if $\min_{\sigma \in \mathcal{S}_k} \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$, where σ is a matching between indices of clusters of $\mathcal{C}'$ and those of $\mathcal{C}$.*

Definition 4. *A clustering instance (P, d) is (α, ϵ)-**perturbation resilient** to a given objective Φ if for any function $d' : P \times P \to \mathbf{R}_{\geq 0}$ s.t. $\forall p, q \in P, d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$, the optimal clustering $\mathcal{C}'$ for Φ under d' is ϵ-close to the optimal clustering $\mathcal{C}$ for Φ under d.*

For $A, B \subseteq P$ we define $d(A, B) := \sum_{p \in A} \sum_{q \in B} d(p, q)$ and $d(p, B) := d(\{p\}, B)$. Also, we define $d_{\mathrm{a}}(A, B) := d(A, B)/(|A||B|)$ and $d_{\mathrm{a}}(p, B) := d_{\mathrm{a}}(\{p\}, B)$. For simplicity, we will sometimes assume that $\min_i |C_i|$ is known. (Otherwise, we can simply search over the n possible different values.)

Finally, if $A \cap B = \emptyset$, we sometimes write $A \cup B$ as $A + B$ to emphasize that they are disjoint.

2.2 α-Perturbation Resilience for Center-based Objectives

In this section we show that, for $\alpha \geq 1 + \sqrt{2}$, if the clustering instance is α-perturbation resilient for center-based objectives, then we can in polynomial time find the optimal clustering. This improves on the $\alpha \geq 3$ bound of [9] and stands in sharp contrast to the **NP**-Hardness results on worst-case instances. Our algorithm succeeds for an even weaker property, the α-center proximity, introduced in [9].

Definition 5. *A clustering instance (P, d) satisfies the α-**center proximity** property if for any optimal cluster $C_i \in \mathcal{C}$ with center c_i, $C_j \in \mathcal{C}(j \neq i)$ with center c_j, any point $p \in C_i$ satisfies $\alpha d(p, c_i) < d(p, c_j)$.*

Lemma 1. *Any clustering instance that is α-perturbation resilient to center-based objectives also satisfies the α-center proximity.*

The proof follows easily by constructing a specific perturbation that blows up all the pairwise distances within cluster C_i by a factor of α. By α-perturbation resilience, the optimal clustering remains the same after this perturbation. This then implies the desired result. The full proof appears in [9]. In the remainder of this section, we prove our results for α-center proximity, but because it is a weaker condition, our upper bounds also hold for α-perturbation resilience.

We begin with some key properties of α-center proximity instances.

Lemma 2. *For any points $p \in C_i$ and $q \in C_j(j \neq i)$ in the optimal clustering of an α-center proximity instance, when $\alpha \geq 1 + \sqrt{2}$, we have: (1) $d(c_i, q) > d(c_i, p)$, (2) $d(p, c_i) < d(p, q)$.*

Proof. (1) Lemma 1 gives us that $d(q, c_i) > \alpha d(q, c_j)$. By the triangle inequality, we have $d(c_i, c_j) \leq d(q, c_j) + d(q, c_i) < (1 + \frac{1}{\alpha}) d(q, c_i)$. On the other hand, $d(p, c_j) > \alpha d(p, c_i)$ and therefore $d(c_i, c_j) \geq d(p, c_j) - d(p, c_i) > (\alpha - 1) d(p, c_i)$. Combining these inequalities, we get (1).

(2) It suffices to prove $d(p,q) > (\alpha-1)\max\{d(p,c_i), d(q,c_j)\}$. The proof first appears in [9], and we include it for completeness. Without loss of generality, we can assume that $d(p,c_i) \geq d(q,c_j)$. By triangle inequality we have $d(p,q) \geq d(p,c_j) - d(q,c_j)$. From Lemma 1 we have $d(p,c_j) > \alpha d(p,c_i)$. Hence $d(p,q) > \alpha d(p,c_i) - d(q,c_j) \geq (\alpha-1)d(p,c_i) \geq (\alpha-1)d(q,c_j)$. □

Lemma 2 implies that for any optimal cluster C_i, the ball of radius $\max_{p\in C_i} d(c_i,p)$ around the center c_i contains *only* points from C_i, and moreover, points inside the ball are each closer to the center than to any point outside the ball. Inspired by this structural property, we define the notion of closure distance between two sets as the radius of the minimum ball that covers the sets and has some margin from points outside the ball. We show that any (strict) subset of an optimal cluster has smaller closure distance to another subset in the same cluster than to any subset of other clusters or to unions of other clusters. Using this, we will be able to define an appropriate linkage procedure that, when applied to the data, produces a tree on subsets that will all be laminar with respect to the clusters in the optimal solution. This will then allow us to extract the optimal solution using dynamic programming applied to the tree.

We now define the notion of closure distance and then present our algorithm for α-center proximity instances (Algorithm 1). Let $\mathbb{B}(p,r) = \{q : d(q,p) \leq r\}$.

Definition 6. *The **closure distance** $d_P(A,A')$ between two disjoint non-empty subsets A and A' of point set P is the minimum $d \geq 0$ such that there is a point $c \in A \cup A'$ satisfying the following requirements:*

(1) coverage: the ball $\mathbb{B}(c,d)$ covers A and A', i.e. $A \cup A' \subseteq \mathbb{B}(c,d)$;

(2) margin: points inside $\mathbb{B}(c,d)$ are closer to the center c than to points outside, i.e. $\forall p \in \mathbb{B}(c,d), q \notin \mathbb{B}(c,d)$, we have $d(c,p) < d(p,q)$.

Note that $d_P(A,A') = d_P(A',A) \leq \max_{p,q\in P} d(p,q)\ \forall A, A'$, and it can be computed

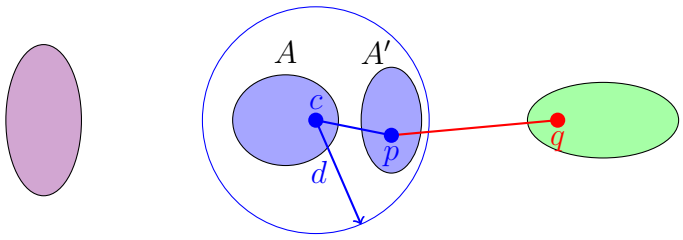


Figure 1: The closure distance.

in polynomial time.

Algorithm 1 Center-based objectives, α perturbation resilience

Input: Data set P, distance function $d(\cdot, \cdot)$ on P.

1: Begin with n singleton clusters.
2: Repeat till only one cluster remains:
merge clusters C, C' which minimize $d_P(C, C')$.
3: Let $\mathcal{T}$ be the tree with single points as leaves and internal nodes corresponding to the merges performed.
4: Run dynamic programming on $\mathcal{T}$ to get the minimum cost pruning $\tilde{\mathcal{C}}$.

Output: Clustering $\tilde{\mathcal{C}}$.

Theorem 1. *For $(1+\sqrt{2})$-center proximity instances, Algorithm 1 outputs the optimal clustering in polynomial time.*

The proof follows immediately from the following key property of Algorithm 1. The details of dynamic programming are presented in Appendix A.1, and an efficient implementation of the algorithm is presented in Appendix A.2.

Theorem 2. *For $(1+\sqrt{2})$-center proximity instances, Algorithm 1 constructs a binary tree $\mathcal{T}$ such that the optimal clustering is a pruning of this tree.*

Proof. We prove correctness by induction. In particular, assume that our current clustering is *laminar* with respect to the optimal clustering – that is, for each cluster A in our current clustering and each C in the optimal clustering, we have either $A \subseteq C$, or $C \subseteq A$ or $A \cap C = \varnothing$. This is clearly true at the start. To prove that the merge steps keep the laminarity, we need to show the following: if A is a

strict subset of an optimal cluster C_i, A' is a subset of another optimal cluster or the union of one or more other clusters, then there exists B from $C_i \setminus A$, such that $d_P(A, B) < d_P(A, A') = d_P(A', A)$.

We first prove that there is a cluster $B \subseteq C_i \setminus A$ in the current cluster list such that $d_P(A, B) \leq d = \max_{p \in C_i} d(c_i, p)$. There are two cases. First, if $c_i \notin A$, then define B to be the cluster in the current cluster list that contains c_i. By induction, $B \subseteq C_i$ and thus $B \subseteq C_i \setminus A$. Then we have $d_P(B, A) \leq d$ since there is $c_i \in B$, and (1) for any $p \in A \cup B$, $d(c_i, p) \leq d$, (2) for any $p \in P$ satisfying $d(c_i, p) \leq d$, and any $q \in P$ satisfying $d(c_i, q) > d$, by Lemma 2 we know $p \in C_i$ and $q \notin C_i$, and thus $d(c_i, p) < d(p, q)$. In the second case when $c_i \in A$, we pick any $B \subseteq C_i \setminus A$ and a similar argument gives $d_P(A, B) \leq d$.

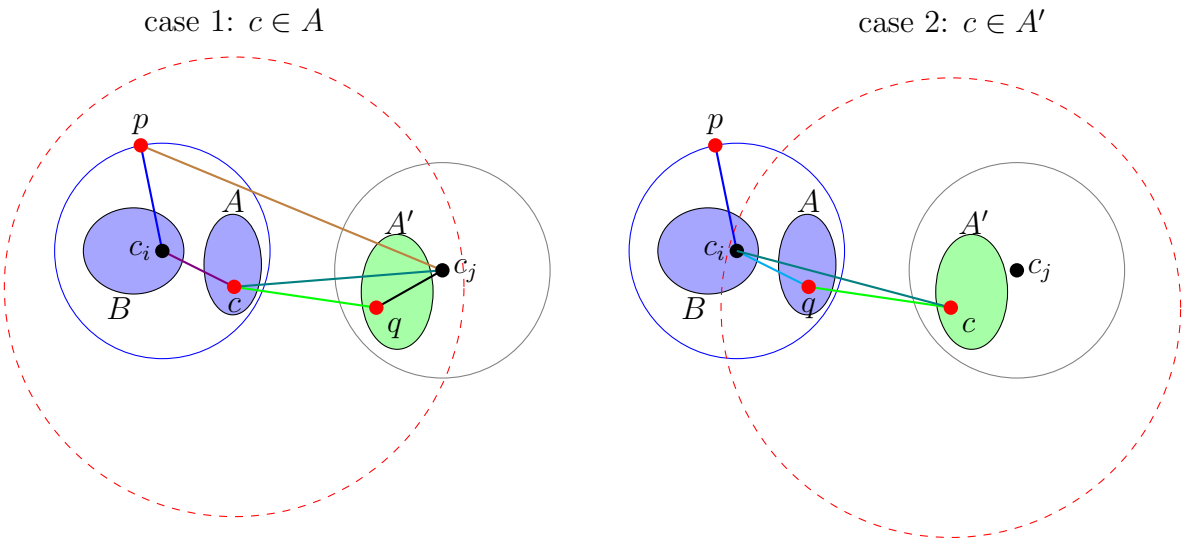


Figure 2: Comparing d and $d_P(A, A')$ in closure linkage.

As a second step, we need to show that $d < \hat{d} = d_P(A, A')$. There are two cases: the center for $d_P(A, A')$ is in A or in A'. See Figure 2 for an illustration. In the first case, there is a point $c \in A$ such that c and $\hat{d}$ satisfy the requirements of the closure distance. Pick a point $q \in A'$, and define C_j to be the cluster in the optimal

clustering that contains q. As $d(c,q) \leq \hat{d}$, and by Lemma 2 $d(c_j,q) < d(c,q)$, we must have $d(c_j,c) \leq \hat{d}$ (otherwise it violates the second requirement of closure distance). Suppose $p = \arg\max_{p' \in C_i} d(c_i,p')$. Then we have $d = d(p,c_i) < d(p,c_j)/\alpha \leq (d + d(c_i,c) + d(c,c_j))/\alpha$ where the first inequality comes from Lemma 1 and the second from the triangle inequality. Since $d(c_i,c) < d(c,c_j)/\alpha$, we can combine the above inequalities and compare d and $d(c,c_j)$, and when $\alpha \geq 1+\sqrt{2}$ we have $d < d(c,c_j) \leq \hat{d}$.

Now consider the second case, when there is a point $c \in A'$ such that c and $\hat{d}$ satisfy the requirements in the definition of the closure distance. Select an arbitrary point $q \in A$. We have $\hat{d} \geq d(c,q)$ from the first requirement, and $d(c,q) > d(c_i,q)$ by Lemma 2. Then from the second requirement of closure distance $d(c_i,c) \leq \hat{d}$. And by Lemma 2, $d = d(c_i,p) < d(c_i,c)$, we have $d < d(c_i,c) \leq \hat{d}$. □

Note: Our factor of $\alpha = 1 + \sqrt{2}$ beats the **NP**-hardness *lower bound* of $\alpha = 3$ of [9] for center-proximity instances. The reason is that the lower bound of [9] requires the addition of Steiner points that can act as centers but are not part of the data to be clustered (though the upper bound of [9] does not allow such Steiner points). One can also show a lower bound for center-proximity instances without Steiner points. In particular one can show that for any $\epsilon > 0$, the problem of solving $(2-\epsilon)$-center proximity k-median instances is **NP**-hard [95].

2.3 (α, ϵ)-Perturbation Resilience for the k-Median Objective

In this section we consider a natural relaxation of the α-perturbation resilience, the (α,ϵ)-perturbation resilience property, that requires the optimum after perturbation of up to a multiplicative factor α to be ϵ-close to the original (one should think of ϵ as sub-constant). We show that if the instance is (α,ϵ)-perturbation resilient, with $\alpha > 4$ and $\epsilon = O(\epsilon'\rho)$ where ρ is the fraction of the points in the smallest cluster, then we can in polynomial time output a clustering that provides a $(1+\epsilon')$-approximation to

the optimum. Thus this improves over the best worst-case approximation guarantees known [76] when $\epsilon' \leq \sqrt{3}$ and also beats the lower bound of $(1 + 1/e)$ on the best approximation achievable on worst case instances for the metric k-median objective [55, 59] when $\epsilon' \leq 1/e$.

The key idea is to understand and leverage the structure implied by (α, ϵ)-perturbation resilience. We show that perturbation resilience implies that there exists only a small fraction of points that are bad in the sense that their distance to their own center is not α times smaller than their distance to any other centers in the optimal solution. We then use this bounded number of bad points in our clustering algorithm.

2.3.1 Structure of (α, ϵ)-Perturbation Resilient Instances

To understand (α, ϵ)-perturbation resilience, we need to consider the difference between the optimal clustering $\mathcal{C}$ under d and the optimal clustering $\mathcal{C}'$ under d', defined as $\min_{\sigma \in \mathcal{S}_k} \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}|$. Without loss of generality, we assume in this subsection that $\mathcal{C}'$ is indexed so that the argmin σ is the identity, and the distance between $\mathcal{C}$ and $\mathcal{C}'$ is $\sum_{i=1}^{k} |C_i \setminus C'_i|$. We denote by c'_i the center of C'_i.

In the following we call a point *good* if it is α times closer to its own center than to any other center in the optimal clustering; otherwise we call it *bad*. Let B_i be the set of bad points in C_i. That is, $B_i = \{p \in C_i : \exists j \neq i, \alpha d(c_i, p) > d(c_j, p)\}$. Let $G_i = C_i \setminus B_i$ be the good points in cluster C_i. Let $B = \cup_i B_i$ and $G = \cup_i G_i$. We show that under perturbation resilience we do not have too many bad points. Formally:

Theorem 3. *Suppose the clustering instance is (α, ϵ)-perturbation resilient for the k-median objective and $\min_i |C_i| > (3 + \frac{2\alpha}{\alpha - 1})\epsilon n + 9\alpha$. Then $|B| \leq \epsilon n$.*

The main idea is to construct a specific perturbation that forces certain selected bad points to move from their original optimal clusters. Then the (α, ϵ)-perturbation resilience leads to a bound on the number of selected bad points, which can also be proved to be a bound on all the bad points. The selected bad points $\hat{B}_i$ in cluster i are

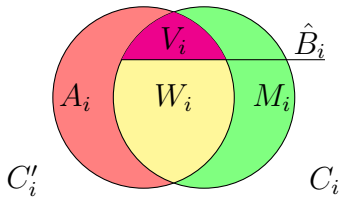


Figure 3: Different types of points when bounding the number of bad points for (α, ϵ)-perturbation resilient k-median instances.

defined by arbitrarily selecting $\min(\epsilon n + 1, |B_i|)$ points from B_i. Let $c(p)$ denote the second nearest center for $p \in \hat{B}_i$ and the nearest center for $p \notin \hat{B}_i$. The perturbation we consider blows up all distances by a factor of α except for those distances between p and $c(p)$. Suppose after perturbation, the optimal cluster C'_i is obtained by adding point set A_i and removing point set M_i from C_i, i.e. $A_i = C'_i \setminus C_i, M_i = C_i \setminus C'_i$. $C_i \cap C'_i$ can be divided into two parts: $W_i = (C_i \cap C'_i) \setminus \hat{B}_i$ and $V_i = (C_i \cap C'_i) \cap \hat{B}_i$. Then we have $C_i = W_i + V_i + M_i, C'_i = W_i + V_i + A_i$. See Figure 3 for an illustration.

The key challenge in proving a bound on the selected bad points is to show that $c'_i = c_i$ for all i. This means the optimal centers do not change after the perturbation. Then in the optimum under d' each point p is assigned to the center $c(p)$, and therefore the selected bad points will move from their original optimal clusters. By (α, ϵ)-perturbation resilience, we get an upper bound on the number of selected bad points.

At a high level, we prove that $c_i = c'_i$ for all i as follows. We first show that for each cluster, its new center is close to its old center, roughly speaking since the new and old cluster have a lot in common (Claim 1). We then show if $c_i \neq c'_i$ for some i, then the weighted sum of the distances $\sum_{1 \leq i \leq k} |C_i| d(c_i, c'_i)$ should be large (Claim 2). However, this contradicts Claim 1, so the centers do not move after the perturbation.

In the following, we will prove the two claims that imply $c'_i = c_i (\forall i)$ and then use them to prove the theorem. The proofs make frequent use of the translation from d' to d, which is summarized in Fact 1.

Fact 1. *Suppose the clustering instance is* (α, ϵ)*-perturbation resilient and* $\min_i |C_i| > (\frac{2}{\alpha-1} + 3)\epsilon n + 1$*. If* $c'_i \neq c_i$*, then we have*

$$d'(c'_i, W_i) \geq \alpha d(c'_i, W_i \setminus \{c(c'_i)\}), \qquad d'(c_i, W_i) = d(c_i, W_i),$$

$$d'(c'_i, V_i) = \alpha d(c'_i, V_i), \qquad d'(c_i, V_i) = \alpha d(c_i, V_i),$$

$$d'(c'_i, A_i) \geq \alpha d(c'_i, A_i \setminus \{c(c'_i)\}), \qquad d'(c_i, A_i) \leq \alpha d(c_i, A_i \setminus \{c(c'_i)\}) + \alpha(1+\alpha) d(c'_i, c_i).$$

Proof Sketch. If $d'(c'_i, C)$ does not involve distances between p and $c(p)$ for any $p \in P$, then $d'(c'_i, C) = \alpha d(c'_i, C)$. To apply this idea, we first need to show that $c'_i \neq c_j (\forall j \neq i)$. Intuitively, if $c'_i = c_j$, then under d', points in W_j should be closer to c'_j than to $c'_i = c_j$. So under d, these points are α time closer to c'_j than to c_j, which means the distance between c_j and c'_j is not so large compared to the average distance between c_j and W_j by the triangle inequality. On the other hand, it also means c_j has $(1 - 1/\alpha) d(c_j, W_j)$ more cost than c'_j on W_j. Then c'_j should have at least $(1-1/\alpha) d(c_j, W_j)$ more cost on $C_j \setminus W_j$. By the triangle inequality, the distance between c_j and c'_j is much larger than the average distance between c_j and W_j, which is contradictory. Therefore, $c'_i \neq c_j$.

Then we only need to check if $c(c'_i) \in C$ when translating $d'(c'_i, C)$ to $d(c'_i, C)$ by a case-by-case study. The same idea can be applied to $d'(c_i, C)$. The formal proof is presented in Appendix A.3. □

Claim 1. *For each* i*,* $d(c_i, W_i) \geq \frac{\alpha+2}{\alpha+1} \frac{|C_i|}{3} d(c_i, c'_i)$.

Proof. The key idea is that under d', c'_i is the optimal center for C'_i, so it has no more cost than c_i on C'_i. Since V_i and A_i are small compared to W_i, c'_i cannot save much cost on V_i and A_i, so it cannot have much more cost on W_i than c_i. Then c'_i is close to W_i (compared to the distance between c_i and W_i). By triangle inequality, c'_i is close to c_i.

Formally, if $c'_i = c_i$, $d(c'_i, c_i) = 0$, which immediately implies the bound. Otherwise, we need to use the fact that c'_i has smaller cost than c_i on C'_i under d': $d'(c'_i, C'_i) \leq$

$d'(c_i, C'_i)$. We divide C'_i into three parts W_i, V_i and A_i, and move terms on W_i to one side (the cost more than c_i on W_i), the rest terms to another side (the cost saved on V_i and A_i):

$$d'(c'_i, W_i) - d'(c_i, W_i) \leq d'(c_i, A_i) - d'(c'_i, A_i) + d'(c_i, V_i) - d'(c'_i, V_i)$$

Translating d' to d by Fact 1, we have

$$\begin{aligned} & \alpha d(c'_i, W_i \setminus \{c(c'_i)\}) - d(c_i, W_i) \\ \leq\ & \alpha d(c_i, A_i \setminus \{c(c'_i)\}) + \alpha(1+\alpha) d(c'_i, c_i) - \alpha d(c'_i, A_i \setminus \{c(c'_i)\}) \\ & + \alpha d(c_i, V_i) - \alpha d(c'_i, V_i). \end{aligned}$$

By the triangle inequality,

$$\alpha d(c_i, c'_i)(|W_i| - 1) - (\alpha + 1) d(c_i, W_i) \leq \alpha d(c_i, c'_i) \left[|V_i| + |A_i| + (1+\alpha)\right]$$

which implies the desired result. □

Claim 2. *Let $I_i = 1$ if $c_i \neq c'_i$ and $I_i = 0$ otherwise. Then $\sum_{1 \leq i \leq k} I_i d(c_i, W_i) \leq \sum_{1 \leq i \leq k} \frac{|C_i|}{3} d(c_i, c'_i)$.*

Proof. The key idea is that the clustering that under d' assigns points in W_i and A_i to c_i and points p in V_i to $c(p)$, saves as much cost as $d'(c'_i, W_i) - d'(c_i, W_i) \approx (\alpha - 1) d(c_i, W_i)$ on W_i compared to the optimal clustering $\{C'_i\}$ under d', if $c'_i \neq c_i$. Then $\{C'_i\}$ must save this cost on other parts of points. So $\{c'_i\}$ should be near these points and $\{c_i\}$ should be far away. By the triangle inequality, the weighted sum of the distances between $\{c'_i\}$ and $\{c_i\}$ should be large.

Formally, we have the following inequality from the fact that $\{c'_i\}$ are the optimal centers under d', thus have no more cost than the clustering that under d' assigns points in $W_i \cup A_i$ to c_i and points p in V_i to $c(p)$:

$$\sum_{i=1}^{k} d'(c'_i, C'_i) \leq \sum_{i=1}^{k} \left[d'(c_i, C'_i \setminus V_i) + \sum_{p \in V_i} d'(c(p), p) \right].$$

We divide C'_i into A_i, V_i and W_i, and divide $C'_i \setminus V_i$ into A_i and W_i. Then we move terms on W_i to one side (the cost more than c_i on W_i), and move the rest terms to another side (the cost saved on V_i and A_i):

$$\sum_{i=1}^{k}[d'(c'_i, W_i) - d'(c_i, W_i)] \leq \sum_{i=1}^{k}\left[d'(c_i, A_i) - d'(c'_i, A_i) + \sum_{p\in V_i} d'(c(p), p) - d'(c'_i, V_i)\right]. \quad (1)$$

To estimate $d'(c'_i, W_i) - d'(c_i, W_i)$, note that $d'(c'_i, W_i) - d'(c_i, W_i) \approx \alpha d(c'_i, W_i) - d(c_i, W_i)$ by Fact 1, so it suffices to show that $d(c_i, W_i)$ is not much larger than $d(c'_i, W_i)$. This follows from the fact that $\{c_i\}$ are the optimal centers under d. Formally, $d(c_i, C_i) \leq d(c'_i, C_i)$, which leads to

$$d(c_i, W_i) - d(c'_i, W_i) \leq d(c'_i, M_i) - d(c_i, M_i) + d(c'_i, V_i) - d(c_i, V_i). \quad (2)$$

Now, we are ready to estimate $d'(c'_i, W_i) - d'(c_i, W_i)$. Multiply Inequality (2) by α and sum over all i, then add it to Inequality (1):

$$\begin{aligned} & \sum_{i=1}^{k}[d'(c'_i, W_i) - d'(c_i, W_i) + \alpha d(c_i, W_i) - \alpha d(c'_i, W_i)] \\ \leq & \sum_{i=1}^{k}\Big[d'(c_i, A_i) - d'(c'_i, A_i) + \alpha d(c'_i, M_i) - \alpha d(c_i, M_i) \\ & + \sum_{p\in V_i} d'(c(p), p) - d'(c'_i, V_i) + \alpha d(c'_i, V_i) - \alpha d(c_i, V_i)\Big]. \end{aligned}$$

Rewrite it as $\sum_i S_i \leq \sum_i T_i$ where S_i and T_i are the terms related to index i on the two sides respectively. Then the claim follows from the following bounds on S_i and T_i and the fact that $|C_i|/3 \geq |A_i| + |M_i| + \alpha + 2$:

$$S_i \geq (\alpha - 1)I_i d(c_i, W_i) - \alpha d(c_i, c'_i), \qquad T_i \leq \alpha(|A_i| + |M_i| + \alpha + 1)d(c_i, c'_i).$$

It remains to prove the two bounds by a case-by-case study. For S_i, we have three cases: if $c'_i = c_i$, then $S_i = 0$; if $c'_i \neq c_i$ and $c(c'_i) \neq c_i$, then it is $0 + (\alpha - 1)d(c_i, W_i)$; if $c(c'_i) = c_i$, then it is $-\alpha d(c_i, c'_i) + (\alpha - 1)d(c_i, W_i)$. In conclusion, in any case $S_i \geq (\alpha - 1)I_i d(c_i, W_i) - \alpha d(c_i, c'_i)$.

For T_i, in the easy case when $c_i' = c_i$, most terms cancel out, and we get $T_i = \sum_{p\in V_i} d(p, c(p)) - \alpha d(c_i, V_i)$. Then $T_i \le (\alpha - \alpha)d(c_i, V_i) = 0$ since V_i are the selected bad points. If $c_i' \neq c_i$, Fact 1 leads to

$$\begin{aligned} T_i &\le \alpha d(c_i, A_i \setminus \{c(c_i')\}) + \alpha(1+\alpha)d(c_i', c_i) - \alpha d(c_i', A_i \setminus \{c(c_i')\}) \\ &+ \alpha d(c_i', M_i) - \alpha d(c_i, M_i) \\ &+ \sum_{p\in V_i} d(p, c(p)) - \alpha d(c_i', V_i) + \alpha d(c_i', V_i) - \alpha d(c_i, V_i). \end{aligned}$$

The first line is bounded by $\alpha(\alpha + 1 + |A_i|)d(c_i, c_i')$. The second line is bounded by $\alpha d(c_i, c_i')|M_i|$. The third line is bounded by 0, since V_i are the selected bad points and thus $\sum_{p\in V_i} d(p, c(p)) \le \alpha d(c_i, V_i)$. In conclusion, in any case $T_i \le \alpha(|A_i| + |M_i| + \alpha + 1)d(c_i, c_i')$. □

Proof of Theorem 3. Combining Claim 1 and 2 leads to

$$\sum_{1\le i\le k} |C_i| d(c_i, c_i') \left[1 - (\alpha+2)I_i/(\alpha+1)\right] \ge 0.$$

If $I_i = 0$, then $d(c_i, c_i') = 0$; if $I_i = 1$, the coefficient of $d(c_i, c_i')$ is negative. So all terms on the left hand side are equal to 0, i.e. $d(c_i, c_i') = 0 (1 \le i \le k)$. Then points in $\hat{B}_i$ will move to other clusters after perturbation, which means that $\hat{B}_i \subseteq M_i$. So $|\cup_i \hat{B}_i| \le |\cup_i M_i| \le \epsilon n$. In particular, $|\hat{B}_i| \le \epsilon n$ for any i. Then $|B_i| \le \epsilon n$, otherwise $|\hat{B}_i|$ would be $\epsilon n + 1$. Therefore, $\hat{B}_i = B_i$, and $|B| = |\cup_i \hat{B}_i| \le \epsilon n$. □

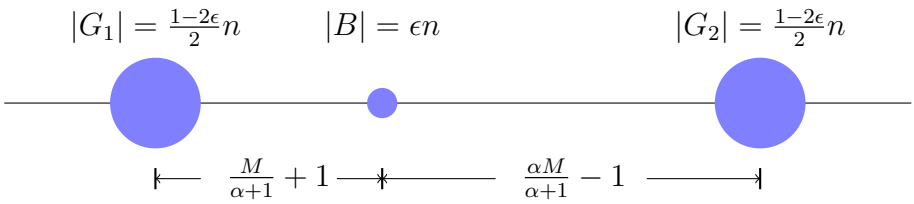


Figure 4: The optimality of the bound on the number of bad points for (α, ϵ)-perturbation resilient k-median instances.

Note 1: The bound in Theorem 3 is optimal in the sense that for any $\alpha > 1, \epsilon < \frac{1}{5}$, we can easily construct an (α, ϵ)-perturbation resilient 2-median instance which has ϵn bad points.

The instance is shown in Figure 4. It has 3 groups of points: G_1, G_2, and B. Both G_1 and G_2 have $(1-\epsilon)n/2$ points, and B has ϵn points. Let M be a sufficiently large constant, say, $M > n^2/\epsilon$. The distances within the same group are 1, while those between the points in G_1 and G_2 are M, those between the points in B and G_1 are $\frac{M}{\alpha+1}+1$, and those between the points in B and G_2 are $\frac{\alpha M}{\alpha+1}-1$. The instance satisfies the triangle inequality, which can be verified by a case-by-case study. The optimal clustering before perturbation has one center in G_1 and the other in G_2. Then B are trivially bad points, and thus we have ϵn bad points in this instance.

Now we show that the instance is (α, ϵ)-perturbation resilient. To prove that the optimal clustering after perturbation $\mathcal{C}'$ is ϵ-close to the original optimal clustering, it suffices to show that $\mathcal{C}'$ has one center from $G_1 \cup B$ and the other center from G_2. Assume for contradiction that this is not true. If both centers come from G_2, the cost of points in G_1 is $\frac{(1-\epsilon)n}{2}M$. On the other hand, the optimal cost before perturbation is $(1-\epsilon)n - 2 + \epsilon n(\frac{M}{\alpha+1}+1)$, so the optimal cost after perturbation is no more than $\alpha[(1-\epsilon)n-2+\epsilon n(\frac{M}{\alpha+1}+1)]$. But this is smaller than $\frac{(1-\epsilon)n}{2}M$, which is a contradiction. Similarly, we get a contradiction if both centers come from $G_1 \cup B$.

Note 2: Theorem 3 requires that the sizes of the optimal clusters are sufficiently large. This makes sure that a majority of points in each optimal cluster stay after moving at most ϵn points, which means W_i is significantly larger than V_i, A_i and M_i. This fact is crucial for proving the theorem. In the following subsections, we assume $n > \epsilon/\alpha$, so that when $\alpha - 1 = O(1)$, the requirement in Theorem 3 is simplified as $\min_i |C_i| = \Omega(\epsilon n)$.

2.3.2 Approximating the Optimum Clustering

Now, we consider approximating the cost of the optimum clustering. We can see that after removing the bad points, the optimal clusters are far apart from each other. In order to get rid of the influence of the bad points, we generate a list of blobs, which form a partition of the data points, and each of which contains only good points from one optimal cluster. Then we construct a tree on the list of blobs with a pruning that assigns all good points correctly. We will show that this pruning has low cost, so the lowest cost pruning of the tree is a good approximation. The details are described in Algorithm 2.

A key step is to generate the list of almost "pure" blobs, which is described in Algorithm 3. Suppose for any i and any good point $p \in G_i$, its $\gamma|G_i|$ nearest neighbors contain no good points outside C_i. Also suppose the algorithm knows the value of γ. Informally, the algorithm maintains a threshold t. At each threshold, for each point p that has not been added to the list, the algorithm checks its γt nearest neighbors $N_{\gamma t}(p)$. It constructs a graph F_t by connecting any two points that have most neighbors in common. It then builds another graph H_t by connecting any two points that have sufficiently many neighbors in F_t, and adds sufficiently large components in H_t to the list. Finally, for each remaining point p, it checks if most of p's neighbors are in the list and if there are blobs containing a significant amount of p's neighbors. If so, it inserts p into such a blob with the smallest median distance. Then the threshold is increased and the above steps are repeated.

The intuition behind Algorithm 3 is as follows. As mentioned above, the algorithm works when for any i and any good point $p \in G_i$, the $\gamma|G_i|$ nearest neighbors of p contain no good points outside C_i ($\gamma = 1$ for the k-median instances considered in this section, as shown in Lemma 3; $\gamma = \frac{4}{5}$ for the min-sum instances considered in Section 2.5, as shown in Claim 5). Without loss of generality, assume $|C_1| \leq |C_2| \leq \cdots \leq |C_k|$. When $t \leq |C_1|$, good points in different clusters do not have

most neighbors in common and thus are not connected in F_t. However, they may be connected by a path of bad points. So we further build the graph H_t to disconnect such paths, which ensures that the blobs added into the list contain only good points from one optimal cluster. The final insert step (Step 6) makes sure that when $t = |C_1|$, all remaining good points in C_1 will be added to the list and will not affect the construction of blobs from other optimal clusters. We can show by induction that, at the end of the iteration $t = |C_i|$, all good points in $C_j (j \leq i)$ are added to the list. When t is large enough, any remaining bad points are inserted into the list, so the points are partitioned into a list of almost pure blobs. The formal guarantee for Algorithm 3 is stated in Lemma 4.

Another key step is to construct a tree on these blobs. Since good points are closer to good points in the same optimal cluster than to those in other clusters (Lemma 3), there exist algorithms that can build a tree with a pruning that assigns all good points correctly. In particular, we can use the robust linkage procedure in [16], which repeatedly merges the two blobs C, C' with the maximum $\text{score}(C, C')$ defined as follows. For each $p \in C$, sort the other blobs in decreasing order of the median distance between p and points in the blob, and let $\text{rank}(p, C')$ denote the rank of C'. Then define $\text{rank}(C, C') = \text{median}_{x \in C}[\text{rank}(x, C')]$ and $\text{score}(C, C') = \min[\text{rank}(C, C'), \text{rank}(C', C)]$. Intuitively, for any blobs A, A' from the same optimal cluster and D from a different cluster, good points in A always rank A' later than D in the sorted list, so $\text{rank}(A, A') > \text{rank}(A, D)$. Similarly, $\text{rank}(A', A) > \text{rank}(A', D)$, and thus $\text{score}(A', A) > \text{score}(A, D)$. This means the algorithm will always merge blobs from the same cluster before merging them with blobs outside, so there is a pruning that assigns all good points correctly.

In the following, we prove that Algorithm 2 outputs a good approximation. We begin with a key property of (α, ϵ)-perturbation resilience that ensures the success of Algorithm 3.

Algorithm 2 k-median, (α, ϵ) perturbation resilience

Input: Data set P, distance function $d(\cdot,\cdot)$ on P, $\min_i |C_i|$, $\epsilon > 0$

1: Run Algorithm 3 to generate a list $\mathcal{L}$ of blobs with parameters $m_B = \epsilon n, \gamma = 1$.
2: Run the robust linkage procedure in [16] to get a cluster tree $\mathcal{T}$.
3: Run dynamic programming on $\mathcal{T}$ to get the minimum cost pruning $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$.

Output: Clustering $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$.

Algorithm 3 Generating interesting blobs

Input: Data set P, distance function $d(\cdot,\cdot)$ on P, the size of the smallest optimal cluster $\min_i |C_i|$, the upper bound on the number of bad points m_B, a parameter $\gamma \in [4/5, 1]$

1: Let $N_r(p)$ denote the r nearest neighbors of p in P.
2: Let $\mathcal{L} = \emptyset, A_P = P$. Let the initial threshold $t = \min_i |C_i|$.
3: Construct a graph F_t by connecting $p, q \in A_P$ if $|N_{\gamma t}(p) \cap N_{\gamma t}(q)| > (2\gamma - 1)t - 2m_B$.
4: Construct a graph H_t by connecting points $p, q \in A_P$ if p, q share more than m_B neighbors in F_t.
5: Add to $\mathcal{L}$ all the components C of H_t with $|C| \geq \frac{1}{2}\min_i |C_i|$ and remove them from A_P.
6: For each point $p \in A_P$, check if most of $N_{\gamma t}(p)$ are in $\mathcal{L}$ and if there exists $C \in \mathcal{L}$ containing a significant number of points in $N_{\gamma t}(p)$. More precisely, check if
(1) $|N_{\gamma t}(p) \setminus \mathcal{L}| \leq \frac{1}{2}\min_i |C_i| + 2m_B$;
(2) $\mathcal{L}_p \neq \emptyset$ where $\mathcal{L}_p = \{C \in \mathcal{L} : |C \cap N_{\gamma t}(p)| \geq (\gamma - \frac{3}{5})|C|\}$.
If so, assign p to the blob in $\mathcal{L}_p$ of smallest median distance, remove p from A_P.
7: While $|A_P| > 0$, increase t by 1 and go to Step 3.

Output: The list $\mathcal{L}$.

Lemma 3. *When $\alpha > 4$, for any good points $p_1, p_2 \in G_i, q \in G_j (j \neq i)$, we have $d(p_1, p_2) < d(p_1, q)$. Consequently, for any good point $p \in G_i$, all its $|G_i|$ nearest neighbors belong to $C_i \cup B$.*

Proof. By the same proof in Lemma 2(2), we have

$$d(p_1, q) > (\alpha - 1)d(p_1, c_i) \qquad \text{and} \qquad d(p_2, q) > (\alpha - 1)d(p_2, c_i).$$

These then lead to

$$d(p_1, p_2) \;\leq\; d(p_1, c_i) + d(p_2, c_i) < \frac{d(p_1, q) + d(p_2, q)}{\alpha - 1} \leq \frac{2d(p_1, q) + d(p_1, p_2)}{\alpha - 1}$$

and thus $d(p_1, p_2) < \frac{2}{\alpha - 2} d(p_1, q)$. When $\alpha > 4$, we have $d(p_1, p_2) < d(p_1, q)$. □

Lemma 4. *Suppose the number of bad points is bounded by m_B, and for any i and any good point $p \in G_i$, all its $\gamma|G_i|$ nearest neighbors in P are from $C_i \cup B$. If $\min_i |C_i| > 30m_B$, then Algorithm 3 generates a list $\mathcal{L}$ of blobs each of size at least $\frac{1}{2}\min_i |C_i|$ such that:*

- *The blobs in $\mathcal{L}$ form a partition of P.*
- *Each blob in $\mathcal{L}$ contains good points from only one optimal cluster.*

Proof. We prove the following two claims by induction on $i \leq k$:

- For any $t \leq |G_i|$, any blob in the list $\mathcal{L}$ only contains good points from only one optimal cluster; all blobs have size at least $\frac{1}{2}\min_i |C_i|$.
- At the beginning of the iteration $t = |G_i| + 1$, any good point $p \in G_j (j \leq i)$ has been assigned to a blob in the list that contains good points only from C_j.

The first two claims imply that each blob in the list contains good points from only one optimal cluster. Moreover, at the beginning of the iteration $t = |G_k| + 1$, all good points have been assigned to one of the blobs in $\mathcal{L}$, so there are only bad points left, the number of which is smaller than $\frac{1}{2}\min_i |C_i|$. These remaining points will eventually be assigned to the blobs before $\gamma t > n$, so the blobs form a partition of P.

The claims are clearly both true initially. We show now that as long as $t \leq |G_1|$, the graphs F_t and H_t have the following properties.

- No good point p_i in cluster C_i is connected in F_t to a good point p_j in a different cluster C_j. By assumption, p_i has no neighbors outside $C_i \cup B$ and p_j has no neighbors outside $C_j \cup B$, so they share at most $m_B < (2\gamma - 1)t - 2m_B$ neighbors.
- No point q is connected in F_t to both a good point p_i in C_i and a good point p_j in a different cluster C_j. If q is connected to p_i, then $|N_{\gamma t}(p_i) \cap N_{\gamma t}(q)| >$

$(2\gamma-1)t-2m_B$. Since p_i has no neighbors outside $C_i \cup B$, $N_{\gamma t}(q)$ contains more than $(2\gamma-1)t-3m_B \geq \gamma t/2$ points from G_i. Similarly, if q is connected to p_j, then $N_{\gamma t}(q)$ contains more than $\gamma t/2$ points from G_j, which is contradictory.

- All the components in H_t of size at least $\frac{1}{2}\min_i |C_i|$ will only contain good points from one optimal cluster. As there are at most m_B bad points, any two points connected in H_t must be connected in F_t to one good point. Then by the above two properties, points on a path in H_t must be connected in F_t to good points in the same cluster, so there is no path connecting good points from different clusters.

We can use the three properties to argue the first claim: as long as $t \leq |G_1|$, each blob in $\mathcal{L}$ contains good points from at most one optimal cluster. This is true at the beginning and by the third property, for any $t \leq |G_1|$, anytime we insert a whole new blob in the list in Step 5, that blob must contain point from at most one optimal cluster. We now argue that this property is never violated as we assign points to blobs already in the list in Step 6. Suppose a good point $p \in C_i$ is inserted into $C \in \mathcal{L}$. Then $C \in \mathcal{L}_p$, which means $|N_{\gamma t}(p) \cap C| \geq |C|/2 > m_B$. So $N_{\gamma t}(p) \cap C$ contains at least one good point, which must be from C_i since $N_{\gamma t}(p)$ contains no good points outside C_i. Then by induction C must contain only good points from C_i, and thus adding p to C does not violate the first claim.

We now show the second claim: after the iteration $t = |G_1|$, all the good points in C_1 have already been assigned to a blob in the list that only contains good points from C_1. There are two cases. First, if at the beginning of the iteration $t = |G_1|$, there are still at least $\frac{1}{2}\min_i |C_i|$ points from the good point set G_1 that do not belong to blobs in the list. Any such good point has all $\gamma|G_1|$ neighbors in $C_1 \cup B$. Then any two such good points share at least $2\gamma|G_1| - |C_1 \cup B| \geq (2\gamma-1)|G_1| - |B| \geq (2\gamma-1)t - 2m_B$ neighbors. So they will connect to each other in F_t and then in H_t, and thus we

will add one blob to $\mathcal{L}$ containing all these points. Second, it could be that at the beginning of the iteration $t = |G_1|$, all but less than $\frac{1}{2}\min_i |C_i|$ good points in G_1 have been assigned to a blob in the list. Denote these points as E. Any point $p \in E$ has no neighbors outside $C_1 \cup B$. Then $|N_{\gamma t}(p) \setminus \mathcal{L}| \leq |E| + |B| \leq \frac{1}{2}\min_i |C_i| + 2m_B$. Also, there exists a blob C containing good points from C_1 such that $C \in \mathcal{L}_p$. Otherwise, $N_{\gamma t}(p)$ contains at most $(\gamma - \frac{3}{5})(|C_1 \cup B|) < \gamma|C_1| - \frac{1}{2}|C_1| - 2m_B$ points in $C_1 \cap \mathcal{L}$, while it contains at most $|E|$ good points in $C_1 \setminus \mathcal{L}$ and contains no points outside $C_1 \cup B$. In total, $N_{\gamma t}(p)$ has less than γt points, which is contradictory. So $\mathcal{L}_p \neq \emptyset$ and p will be added to the list in Step 6.

We then iterate the argument on the remaining set A_P. The key point is that for $t \geq |G_i|, i > 1$, we have that all the good points in $C_1, C_2, \ldots, C_i$ have already been assigned to blobs in $\mathcal{L}$. □

Lemma 3 and 4 show that Algorithm 3 with parameters $m_B = \epsilon n$ and $\gamma = 1$ produces a list of sufficiently large, almost pure blobs. Then the robust linkage procedure in [16] can build a tree on these blobs with a pruning that assigns all good points correctly. Now it suffices to show that this pruning is a good approximation, for which we need to bound the cost increased by the bad points assigned incorrectly. The following property of these bad points turns out to be useful. Intuitively, Algorithm 3 is designed such that whenever a bad point is added to a blob containing good points from a different cluster, it must be closer to a significantly amount of points in that cluster than to a significantly amount of points in its own cluster. Then the cost increased by incorrectly assigning each such bad point is small, resulting in a good approximation.

Lemma 5. *Suppose for any good point $p \in G_i$, all its $|G_i|$ nearest neighbors in P are from $C_i \cup B$, and $\min_i |C_i| > 30m_B$. When running Algorithm 3 with $\gamma = 1$, if a bad point $q \in B_i$ is assigned to a blob C containing good points from a different optimal*

clustering C_j, then there exist $m = \frac{1}{5}\min_i |C_i|$ points Z_i from C_i, and m points Z_j from C_j, such that $d(q, Z_i) \geq d(q, Z_j)$.

Proof. There are two cases: q is added into C in (1) Step 5 or (2) Step 6.

Case 1 There must be a path in H_t connecting q to a good point in C_j at threshold t. For any edge (x, y) in H_t, since x, y share at least ϵn neighbors in F_t and there are at most ϵn bad points, they share at least one good point as neighbor in F_t. As shown in the proof of Lemma 4, no point can connect to good points from different clusters, so in F_t all points on the path must connect to good points in C_j. In particular, q is connected in F_t to a good point $p \in G_j$. Then $|N_t(p) \cap N_t(q)| > t - 2m_B$. Since p is still in A_P, $t \leq |G_j|$, and thus $N_t(p)$ contains no points outside $C_j \cup B$. This means that at least $t - 3m_B \geq m$ points in $N_t(q)$ are good points in C_j, then we can select m points Z_j from $N_t(q) \cap G_j$. We also have that at most $2m_B$ points in $N_t(q)$ are points in C_i, so we can select m points Z_i from $C_i \setminus N_t(q)$.

Case 2 There are three subcases when q is inserted into C at threshold t.

(1) There is no good points from C_i in the list. Since $|N_t(q) \setminus \mathcal{L}| \leq \frac{1}{2}\min_i |C_i| + 2m_B$, $N_t(q)$ contains at most this number of good points in C_i. This means at least $\frac{1}{2}\min_i |C_i| - 2m_B > m$ good points in C_i are outside $N_t(q)$, from which we can select Z_i. On the other hand, we can select Z_j as follows. When inserting q into C, we have $|N_t(q) \cap C| \geq \frac{2}{5}|C| \geq m + m_B$. Since C contains only good points from C_j and some bad points, $N_t(q) \cap C$ contains at least m good points in C_j, from which we can select Z_j. Since Z_j are from $N_t(q)$ and Z_i are outside $N_t(q)$, we have $d(q, Z_i) \geq d(q, Z_j)$.

(2) There exists $C' \in \mathcal{L}$ containing good points from C_i, but $C' \notin \mathcal{L}_p$. This means $|B(q, t) \cap C'| \leq \frac{2}{5}|C'|$, so there are at least $\frac{3}{5}|C'| \geq m + m_B$ points in C' are outside $N_t(q)$. At least m of these points are good points from C_i, since C' contains only good points from C_i and at most m_B bad points. On the other hand, we can select Z_j as in the first subcase.

(3) There exists $C' \in \mathcal{L}_p$ containing good points from C_i. Since q is assigned to C

rather than C' according to median distances, we know that at least half of the points Z'_j from C are closer to q than at least half of the points Z'_i from C'. Since there are at most m_B bad points, we can select m good points Z_j from Z'_j and select m good points Z_i from Z'_i. Note that Z_j are all from G_j and Z_i are all from G_i, so $d(q, Z_i) \geq d(q, Z_j)$. □

Theorem 4. *If the clustering instance is (α, ϵ)-perturbation resilient for $\alpha > 4$ and $\epsilon \leq \rho/30$ where $\rho = \frac{\min_i |C_i|}{n}$, then Algorithm 2 produces a clustering which is $(1 + \frac{5\epsilon}{\rho})$-approximation to the optimal clustering with respect to the k-median objective in polynomial time.*

Proof. By Lemma 3 and 4, Algorithm 3 partition the points into a list of blobs, each of which has size at least $\frac{1}{2}\min_i |C_i|$ and contains only good points from one optimal cluster. Let B'_i denote the bad points that are assigned to blobs containing good points in C_i. By Lemma 3, Theorem 9 in [16] can be applied to $\mathcal{L}$, by which we know that $\{(C_i \cap G) \cup B'_i\}$ is a pruning of the tree. Suppose the cost of the optimum is $\mathcal{OPT}$. We now show that this pruning, using the original centers $\{c_i\}$, is a $(1 + \frac{5\epsilon}{\rho})$-approximation to $\mathcal{OPT}$.

Suppose a bad point $q \in C_i$ is assigned to a blob C containing good points from a different optimal cluster C_j. By Lemma 5, there exist $m = \frac{1}{5}\min_i |C_i|$ points Z_i from C_i, and m points Z_j from C_j, such that $d(q, Z_i) \geq d(q, Z_j)$. Then the increase in cost due to q is bounded as follows:

$$\begin{aligned} d(q, c_j) - d(q, c_i) &\leq \frac{d(q, Z_j) + d(c_j, Z_j)}{m} - \frac{d(q, Z_i) - d(c_i, Z_i)}{m} \\ &\leq \frac{1}{m}[d(c_j, Z_j) + d(c_i, Z_i)] \leq \frac{\mathcal{OPT}}{m}. \end{aligned}$$

As there are at most ϵn bad points and $m = \frac{\min_i |C_i|}{5}$, the increase of cost is at most $\frac{\epsilon n}{m}\mathcal{OPT} = \frac{5\epsilon}{\rho}\mathcal{OPT}$.

Running Time In Algorithm 3, for each $p \in P$, we first sort all the other points in ascending order of distances in time $O(n^2 \log n)$. At each threshold t, think of

a directed t-regular graph E_t, where, for each point q in the t nearest neighbors of a point p, there is a directed edge from p to q in E_t. Let A_E denote the adjacent matrix for E_t, and let $N = A_E A_E^T$. Then N_{pq} is the number of common neighbors between p and q, which can be used in constructing F_t. Computing N takes time $O(n^\omega)$, the state of the art for matrix multiplication. The same method can be used to compute the number of common neighbors in F_t and construct H_t. Since there are $O(n)$ thresholds, the total time for constructing F_t and H_t is $O(n^{\omega+1})$. For the other steps, adding a blob takes time $O(n^2)$ and inserting a point takes time $O(n^2)$. These steps can be performed at most $O(n)$ times, so they take $O(n^3)$ time. In total, Algorithm 3 takes time $O(n^{\omega+1})$. Since the robust linkage algorithm takes time at most $O(n^{\omega+1})$ ([16]), and the dynamic programming takes time $O(n^3)$ (see Appendix A.1), the running time of Algorithm 2 is $O(n^{\omega+1})$. □

We observe that for $\alpha > 4$ we can tolerate ϵ up to approximately ρ (recalling that ρ is the fraction of points in the smallest cluster), and beat the lower bound of $(1+1/e)$ on the best approximation achievable on worst case instances for the metric k-median objective [55, 59] when $\epsilon < \frac{\rho}{5e}$.

2.3.3 Sublinear Time Algorithm for the k-Median Objective

Consider a clustering instance (X, d) that is (α, ϵ)-perturbation resilient to k-median. For simplicity, suppose the distances are normalized to $[0, 1]$. Let $N = |X|$. Let $\rho = \min_i |C_i|/N$ denote the fraction of the points in the smallest cluster, $\zeta = \Phi_X(\mathbf{c})/N$ denote the average cost of the points in the optimum clustering.

Theorem 5. *Suppose (X, d) is (α, ϵ)-perturbation resilient for $\alpha > 4$, $\epsilon < \rho/100$. Then with probability $\geq 1 - \delta$, we can get an implicit clustering that is $2(1 + \frac{16\epsilon}{\rho})$-approximation in time poly$(\log \frac{N}{\delta}, k, \frac{1}{\epsilon}, \frac{1}{\zeta})$.*

Intuition. The main idea is to run Algorithm 2 on a sufficiently large sample P to obtain the minimum cost pruning and the centers $\tilde{\mathbf{c}}$. Then the implicit clustering of

the whole space X assigns each point in X to its nearest center in $\tilde{\mathbf{c}}$. To show that this is a good approximation, it suffices to show that $\Phi_P(\tilde{\mathbf{c}})$ is close to $\Phi_P(\mathbf{c})$ where $\mathbf{c}$ are the optimal centers for X. Note that Algorithm 2 builds a tree with a pruning $\mathcal{P}'$ that assigns all good points correctly. The key is to use the cost of this pruning as a bridge for $\Phi_P(\tilde{\mathbf{c}})$ and $\Phi_P(\mathbf{c})$: on one hand, $\Phi_P(\tilde{\mathbf{c}})$ is no more than the cost of $\mathcal{P}'$ since $\tilde{\mathbf{c}}$ is the centers in the minimum cost pruning; on the other hand, the cost of $\mathcal{P}'$ is roughly bounded by twice $\Phi_P(\mathbf{c})$ by triangle inequality.

Proof. We sample a set P of size $n = \Theta(\frac{k}{\epsilon^2\zeta^2} \ln \frac{N}{\delta})$ and run Algorithm 2 on P to obtain the minimum cost pruning $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$. The implicit clustering of the whole space X then assigns each point in X to its nearest neighbor in $\tilde{\mathbf{c}}$. Recall that if we partition A into $\mathcal{P}$, the cost using centers $\mathbf{p}$ is denoted as $\Phi_A(\mathcal{P}, \mathbf{p})$. If we partition A by assigning points to nearest centers in $\mathbf{p}$, the cost is denoted as $\Phi_A(\mathbf{p})$. We will show that the cost of implicit clustering $\Phi_X(\tilde{\mathbf{c}})$ approximates the optimum $\Phi_X(\mathbf{c})$.

First, we will prove that when n is sufficiently large, with high probability, $\Phi_X(\tilde{\mathbf{c}})/N \approx \Phi_P(\tilde{\mathbf{c}})/n$ and $\Phi_X(\mathbf{c})/N \approx \Phi_P(\mathbf{c})/n$. Formally, for every set of centers $\mathbf{p}$, if $n = \Theta(\frac{k}{\upsilon^2\zeta^2} \log \frac{N}{\delta})$ where $0 < \upsilon < 1$, then

$$\Pr\left[\left|\frac{\Phi_P(\mathbf{p})}{n} - \frac{\Phi_X(\mathbf{p})}{N}\right| > \upsilon \frac{\Phi_X(\mathbf{p})}{N}\right] \leq 2\exp\{-2\upsilon^2\zeta^2 n\} \leq \frac{\delta}{4N^k}.$$

By the union bound, we have with probability at least $1 - \delta/4$, $(1-\upsilon)\Phi_X(\tilde{\mathbf{c}})/N \leq \Phi_P(\tilde{\mathbf{c}})/n$ and $\Phi_P(\mathbf{c})/n \leq (1+\upsilon)\Phi_X(\mathbf{c})/N$. We can choose $\upsilon = \epsilon/20$, then it is sufficient to show $\Phi_P(\tilde{\mathbf{c}}) \leq 2(1 + \frac{12\epsilon}{\rho})\Phi_P(\mathbf{c})$.

Next, since $\tilde{\mathcal{C}}$ may be different from $\mathcal{C} \cap P$, we need to find a bridge for comparing $\Phi_P(\tilde{\mathbf{c}})$ and $\Phi_P(\mathbf{c})$. Now, we turn to analyze Algorithm 2 on P to find such a bridge. First, we know that X has at most ϵN bad points. Since n is sufficiently large, with probability at least $1 - \delta/4$, P has at most $2\epsilon n$ bad points. Similarly, with probability at least $1-\delta/4$, for any i, $|C_i \cap P| > 60\epsilon n$. These ensure that Algorithm 2 can successfully produce a tree with a pruning $\mathcal{P}'$ that assigns all good points in P

correctly, as shown in Theorem 4. Suppose in P, $\mathbf{c}'$ are the optimal centers for $\mathcal{P}'$. Then we can use $\Phi_P(\mathcal{P}', \mathbf{c}')$ as a bridge for comparing $\Phi_P(\tilde{\mathbf{c}})$ and $\Phi_P(\mathbf{c})$.

On one hand, $\Phi_P(\tilde{\mathbf{c}}) \leq \Phi_P(\tilde{\mathcal{C}}, \tilde{\mathbf{c}}) \leq \Phi_P(\mathcal{P}', \mathbf{c}')$. The first inequality comes from the fact that in $\Phi_P(\tilde{\mathbf{c}})$ each point is assigned to its nearest center and the second comes from that $\tilde{\mathcal{C}}$ is the minimum cost pruning.

On the other hand, $\Phi_P(\mathcal{P}', \mathbf{c}') \leq 2\Phi_P(\mathcal{P}', \mathbf{c}) \leq 2(1 + \frac{12\epsilon}{\rho})\Phi_P(\mathbf{c})$. The second inequality comes from an argument similar to that in Theorem 4 and the fact that $\Phi_P(\mathcal{P}', \mathbf{c})$ is different from $\Phi_P(\mathbf{c})$ only on the bad points. The first inequality comes from the triangle inequality. More precisely, for any $N_i' \in \mathcal{P}'$,

$$\begin{aligned} 2|N_i'| \sum_{p \in N_i'} d(p, c_i) &= \sum_{q \in N_i'} \sum_{p \in N_i'} [d(p, c_i) + d(q, c_i)] \geq \sum_{p \in N_i'} \sum_{q \in N_i'} d(p, q) \\ &\geq \sum_{p \in N_i'} \sum_{q \in N_i'} d(q, c_i') = |N_i'| \sum_{q \in N_i'} d(q, c_i') \end{aligned}$$

where the third step follows from the fact that c_i' is the optimal center for N_i'. □

Note: If we have an oracle that given a set of points C_i' finds the best center *in* X for that set, then we can save a factor of 2 in the bound.

2.4 α-Perturbation Resilience for the Min-Sum Objective

In this section we provide an efficient algorithm for clustering α-perturbation resilient instances for the metric min-sum k-clustering problem (Algorithm 4).

Algorithm 4 min-sum, α perturbation resilience

Input: Data set P, distance function $d(\cdot, \cdot)$ on P, $\min_i |C_i|$.

1: Connect each point with its $\frac{1}{2}\min_i |C_i|$ nearest neighbors.
2: Initialize the clustering $\mathcal{C}'$ with each connected component being a cluster.
3: Repeat till only one cluster remains in $\mathcal{C}'$:
merge clusters C, C' in $\mathcal{C}'$ which minimize $d_{\mathrm{a}}(C, C')$.
4: Let $\mathcal{T}$ be the tree with components as leaves and internal nodes corresponding to the merges performed.
5: Run dynamic programming on $\mathcal{T}$ to get the minimum min-sum cost pruning $\tilde{\mathcal{C}}$.

Output: $\tilde{\mathcal{C}}$.

Theorem 6. *For $(3\frac{\max_i |C_i|}{\min_i |C_i|})$-perturbation resilient instances, Algorithm 4 outputs the optimal min-sum k-clustering in polynomial time.*

Intuition. To prove the theorem, first we show that the α-perturbation resilience property implies the following (Lemma 6): for any two different optimal clusters C_i and C_j and any $A \subseteq C_i$, we have $\alpha d(A, C_i \setminus A) < d(A, C_j)$. This follows by considering the perturbation where $d'(p,q) = \alpha d(p,q)$ if $p \in A, q \in C_i \setminus A$ and $d'(p,q) = d(p,q)$ otherwise, and using the fact that the optimum does not change after the perturbation. This can be used to show that when $\alpha > 3\frac{\max_i |C_i|}{\min_i |C_i|}$, we have the following (Lemma 7): (1) for any optimal clusters C_i and C_j and any $A_i \subseteq C_i$, $A_j \subseteq C_j$ such that $\min(|C_i \setminus A_i|, |C_j \setminus A_j|) > \min_i |C_i|/2$ we have $d_{\mathrm{a}}(A_i, A_j) > \min\{d_{\mathrm{a}}(A_i, C_i \setminus A_i), d_{\mathrm{a}}(A_j, C_j \setminus A_j)\}$; (2) for any point p in the optimal cluster C_i, twice its average distance to points in $C_i \setminus \{p\}$ is smaller than the distance to any point in other optimal cluster C_j. Claim (2) implies that for any point $p \in C_i$ its $|C_i|/2$ nearest neighbors are in the same optimal cluster, so the leaves of the tree $\mathcal{T}$ are laminar to the optimum clustering. Claim (1) can be used to show that the merge steps preserve the laminarity with the optimal clustering, so the minimum cost pruning of $\mathcal{T}$ will be the optimal clustering, as desired.

We now prove the key lemmas mentioned above, and then present the detailed proof for the theorem.

Fact 2. *For any nonempty set A and $D \subseteq C$, we have $|D|d(A,C) \leq |C|d(A,D) + |A|d(D, C \setminus D)$.*

Proof. By the triangle inequality, $d_{\mathrm{a}}(A, C \setminus D) \leq d_{\mathrm{a}}(A, D) + d_{\mathrm{a}}(D, C \setminus D)$. The fact then follows from $d(A,C) = d(A,D) + d(A, C \setminus D)$. □

Lemma 6. *Suppose the clustering instance is α-perturbation resilient to the min-sum objective. For any two different optimal clusters C_i and C_j and any $A \subseteq C_i$, we have $\alpha d(A, C_i \setminus A) < d(A, C_j)$.*

Proof. This follows by considering a specific perturbation and using the fact that the optimum does not change after the perturbation. We define a perturbation as follows: $d'(p,q) = \alpha d(p,q)$ if $p \in A, q \in C_i \setminus A$ or $q \in A, p \in C_i \setminus A$, and $d'(p,q) = d(p,q)$ otherwise. d' is a valid α-perturbation of d, so the optimal clustering after perturbation should remain the same. Specially, its cost should be smaller than that of the clustering obtained by replacing C_i, C_j with $C_i \setminus A, A \cup C_j$. After canceling the terms common in the two costs, we have $2d'(A, C_i \setminus A) < 2d'(A, C_j)$, which implies $\alpha d(A, C_i \setminus A) < d(A, C_j)$. □

Lemma 7. *Suppose the clustering instance is α-perturbation resilient to min-sum for $\alpha > 3\frac{\max_i |C_i|}{\min_i |C_i|}$.*

(1) For any two different optimal clusters C_i and C_j and any $A_i \subseteq C_i, A_j \subseteq C_j$, if $|C_i \setminus A_i|$ and $|C_j \setminus A_j|$ are larger than $\min_i |C_i|/2$, then

$$d_{\mathrm{a}}(A_i, A_j) > \min[d_{\mathrm{a}}(A_i, C_i \setminus A_i), d_{\mathrm{a}}(A_j, C_j \setminus A_j)].$$

(2) For any point p, all its $\min_i |C_i|/2$ nearest neighbors are in the same optimal cluster.

Proof. (1) Let $\overline{A}_i = C_i \setminus A_i, \overline{A}_j = C_j \setminus A_j$. We have from Lemma 6 and Fact 2:

$$\alpha d(A_i, \overline{A}_i) < d(A_i, C_j) \leq \frac{1}{|A_j|}\left[|C_j| d(A_i, A_j) + |A_i| d(A_j, \overline{A}_j)\right], \tag{3}$$

$$\alpha d(A_j, \overline{A}_j) < d(A_j, C_i) \leq \frac{1}{|A_i|}\left[|C_i| d(A_j, A_i) + |A_j| d(A_i, \overline{A}_i)\right]. \tag{4}$$

Divide Inequality (3) by $|A_i|$, divide Inequality (4) by $|A_j|$, and add them up:

$$(\alpha - 1)|\overline{A}_j| d_{\mathrm{a}}(A_j, \overline{A}_j) + (\alpha - 1)|\overline{A}_i| d_{\mathrm{a}}(A_i, \overline{A}_i) \;\; < \;\; (|C_i| + |C_j|) d_{\mathrm{a}}(\overline{A}_i, \overline{A}_j).$$

Since α, $|\overline{A}_j|$ and $|\overline{A}_i|$ are large enough, $(\alpha - 1)|\overline{A}_i| > |C_i|$ and $(\alpha - 1)|\overline{A}_j| > |C_j|$. Then the claim follows.

(2) Suppose p comes from the optimal cluster C_i. Let $q = \arg\min_{p' \notin C_i} d(p, p')$, and suppose $q \in C_j$.

If $d_{\mathrm{a}}(p, C_i) \geq d_{\mathrm{a}}(q, C_j)$, then by Inequality (3),

$$\begin{aligned} \alpha d(p, C_i) &\leq |C_j| d(p, q) + d(q, C_j) = |C_j| d(p, q) + |C_j| d_{\mathrm{a}}(q, C_j) \\ &\leq |C_j| d(p, q) + |C_j| d_{\mathrm{a}}(p, C_i) \end{aligned}$$

which leads to $d_{\mathrm{a}}(p, C_i) < d(p, q)/2$ since α is sufficiently large.

If $d_{\mathrm{a}}(p, C_i) < d_{\mathrm{a}}(q, C_j)$, then we have $d_{\mathrm{a}}(q, C_j) < d(p, q)/2$ by a similar argument.

In conclusion, we always have $d_{\mathrm{a}}(p, C_i) < d(p, q)/2$, which means that more than $|C_i|/2$ points in C_i are within distance less than $d(p, q) = \min_{p' \notin C_i} d(p, p')$. □

We are now ready to use Lemma 6 and Lemma 7 to prove the correctness of our theorem.

Proof of Theorem 6. It is sufficient to show that in Algorithm 4, $\mathcal{C}'$ is always laminar to the optimal clustering $\mathcal{C}$, that is, for any $A \in \mathcal{C}'$ and $C \in \mathcal{C}$, we have either $A \subseteq C$, or $C \subseteq A$, or $A \cap C = \varnothing$. Then the minimum cost pruning of $\mathcal{T}$ will be the optimal clustering, which can be obtained by dynamic programming.

Intuitively, Lemma 7(2) implies that $\mathcal{C}'$ is laminar initially, and Lemma 7(1) can be used to show that the merge steps preserve the laminarity, so $\mathcal{C}'$ is always laminar to the optimal clustering.

Formally, we prove the laminarity by induction. By Lemma 7(2), $\mathcal{C}'$ is laminar initially. It is sufficient to prove that if the current clustering is laminar, then the merge step keeps the laminiarity. Assume that our current clustering $\mathcal{C}'$ is laminar to the optimal clustering. Consider a merge of two clusters A and A'. There are two cases when laminarity could fail to be satisfied after the merge: (1) they are strict subsets from different optimal clusters, i.e. $A \subsetneq C_i, A' \subsetneq C_j \neq C_i$; (2) A is a strict subset of an optimal cluster C_i and A' is the union of one or several other optimal cluster(s). By Lemma 7(1), the first case cannot happen. In the second case, for any E that is

a subset of $C_i \setminus A$ in the current clustering, we have $d_{\mathrm{a}}(A, E) \geq d_{\mathrm{a}}(A, A')$. We know that $d_{\mathrm{a}}(A, C_i \setminus A)$ is a weighted average of the average distances between A and the clusters that are subsets of $C_i \setminus A$ in the current clustering, so $d_{\mathrm{a}}(A, C_i \setminus A) \geq d_{\mathrm{a}}(A, A')$. Also, $d_{\mathrm{a}}(A, A')$ is a weighted average of the average distances between A and the optimal clusters in A', so there must exist an optimal cluster $C_j \subseteq A'$ such that $d_{\mathrm{a}}(A, C_j) \leq d_{\mathrm{a}}(A, A') \leq d_{\mathrm{a}}(A, C_i \setminus A)$. This means

$$d(A, C_j) \leq \frac{|C_j|}{|C_i \setminus A|} d(A, C_i \setminus A) \leq \alpha d(A, C_i \setminus A)$$

where the last inequality comes from $\alpha \geq 3\frac{\max_i |C_i|}{\min_i |C_i|}$ and $|C_i \setminus A| \geq \min_i |C_i|/2$. This contradicts Lemma 6. So the merge of the two clusters A and A' will preserve the laminarity.

Running Time Finding the nearest neighbors for each point takes $O(n \log n)$ time, so the step of constructing components takes $O(n^2 \log n)$ time. To compute average distances between clusters, we can record the size of each cluster, and $d(C'_i, C'_j)$ for any C'_i, C'_j in the current clustering, and update $d(C'_i \cup C'_j, C'_l) = d(C'_i, C'_l) + d(C'_j, C'_l)$ for any other cluster C'_l when merging C'_i and C'_j. So the merge steps take $O(n^3)$ time. As dynamic programming takes $O(n^3)$ time, we can find the optimum clustering in $O(n^3)$ time. □

2.4.1 Sublinear Algorithm for the Min-Sum Objective

Here we provide a sublinear algorithm for a clustering instance (X, d) that is α-perturbation resilient to the min-sum objective. For simplicity, suppose the distances are normalized to $[0, 1]$. Let $N = |X|$. Let $\rho = \frac{\min_i |C_i|}{N}$ denote the fraction of the points in the smallest optimal cluster, and $\eta = \min_{p \in X, 1 \leq i \leq k} d_{\mathrm{a}}(p, C_i)$ denote the minimum average distance between points and optimal clusters.

Our main result in this subsection is the following.

Theorem 7. *Suppose the clustering instance (X, d) is α-perturbation resilient to*

Algorithm 5 min-sum, α perturbation resilience, sublinear

Input: Data set X, distance function $d(\cdot,\cdot)$ on X, $\min_i |C_i|$.

1: Draw a sample P of size $n = \Theta(\frac{1}{\rho^2\eta^2} \ln \frac{Nk}{\delta})$ i.i.d. from X.

2: Run Algorithm 4 on P to obtain $\tilde{\mathcal{C}}$.

Output: The implicit clustering of X obtained by assigning each point $p \in X$ to $\tilde{C}_i \in \tilde{\mathcal{C}}$ such that $d(p, \tilde{C}_i)$ is minimized.

the min-sum objective where $\alpha \geq 6\frac{\max_i |C_i|}{\min_i |C_i|}$. *Then with probability at least* $1-\delta$, *Algorithm 5 outputs an implicit optimum clustering in time poly*$(\log \frac{Nk}{\delta}, \frac{1}{\rho\eta})$.

Proof. To prove the theorem, we first show the following (Lemma 8): with high probability, $\mathcal{C}'$ in Algorithm 4 is always laminar to $\mathcal{C} \cap P$. The key idea is that when the sample is sufficiently large, we have that for any $p \in C_i$ and $C_j (j \neq i)$,

$$3\frac{\max_i |C_i \cap P|}{\min_i |C_i \cap P|} d(p, C_i \cap P) < d(p, C_j \cap P)$$

since $\frac{d(p,C_i \cap P)}{n} \approx \frac{d(p,C_i)}{N}$, $\frac{d(p,C_j \cap P)}{n} \approx \frac{d(p,C_j)}{N}$ and $\frac{\max_i |C_i \cap P|}{\min_i |C_i \cap P|} \approx \frac{\max_i |C_i|}{\min_i |C_i|}$. Then $\mathcal{C} \cap P$ satisfies the properties for the linkage in Algorithm 4 to succeed, and thus $\mathcal{C}'$ in Algorithm 4 is always laminar to $\mathcal{C} \cap P$, i.e. $\mathcal{C} \cap P$ is a pruning of the tree.

Then we show that $\mathcal{C} \cap P$ is actually the minimum cost pruning $\tilde{\mathcal{C}}$ (Lemma 9). The key idea is that clusters in $\mathcal{C} \cap P$ are far apart, the cost increased by joining different clusters in it is larger than that saved by splitting clusters, so any other pruning has larger cost than $\mathcal{C} \cap P$. It immediately follows from the two lemmas that the implicit clustering obtained is the optimum clustering $\mathcal{C}$. □

We now present the proofs of the lemmas for the correctness of the theorem.

Lemma 8. *Suppose the clustering instance* (X, d) *is* α*-perturbation resilient to the min-sum objective where* $\alpha \geq 6\frac{\max_i |C_i|}{\min_i |C_i|}$. *If the size of the sample* $n = \Theta(\frac{1}{\rho^2\eta^2} \ln \frac{Nk}{\delta})$, *then with probability at least* $1-\delta$, $\mathcal{C}'$ *in Algorithm 4 is always laminar to* $\mathcal{C} \cap P$.

Proof. The intuition is that on X, for any $i \neq j$, any $p \in C_i$, we have $\alpha d(p, C_i) < d(p, C_j)$. When n is sufficiently large, we can show $d(p, C_i \cap P) \approx \frac{n}{N} d(p, C_i)$ for

any i and $\frac{\max_i |C_i \cap P|}{\min_i |C_i \cap P|} \approx \frac{\max_i |C_i|}{\min_i |C_i|}$, and thus we have a similar claim on P. Then $\mathcal{C}'$ in Algorithm 4 is always laminar to $\mathcal{C} \cap P$.

First, we show that with probability at least $1 - \delta/4$, for any $1 \leq i \leq k$ and $\upsilon = 1/20$,

$$(1 - \upsilon)\frac{n}{N}|C_i| \leq |C_i \cap P| \leq (1 + \upsilon)\frac{n}{N}|C_i|. \tag{5}$$

This follows from the union bound and

$$\Pr\left[\left|\frac{|C_i \cap P|}{n} - \frac{|C_i|}{N}\right| \geq \upsilon\frac{|C_i|}{N}\right] \leq 2\exp\left\{-2\upsilon^2\frac{|C_i|^2}{N^2}n\right\} \leq 2\exp\{-2\upsilon^2\rho^2 n\} \leq \frac{\delta}{4k}.$$

A similar argument shows that with probability at least $1 - \delta/2$, for any $1 \leq i \leq k$ and $p \in X$,

$$(1 - \upsilon)\frac{n}{N}d(p, C_i) \leq d(p, C_i \cap P) \leq (1 + \upsilon)\frac{n}{N}d(p, C_i). \tag{6}$$

Now, by (5), we have $\max_i |C_i \cap P| \leq (1 + \upsilon)\frac{n}{N}\max_i |C_i|$, $\min_i |C_i \cap P| \geq (1 - \upsilon)\frac{n}{N}\min_i |C_i|$. Combined these with (6), we have that with probability at least $1 - \delta$, for any $i \neq j$ and any $p \in C_i$, $3\frac{\max_i |C_i \cap P|}{\min_i |C_i \cap P|}d(p, C_i \cap P) < d(p, C_j \cap P)$, which guarantees the success of Algorithm 4.

□

Lemma 9. *Suppose the clustering instance (X, d) is α-perturbation resilient to the min-sum objective where $\alpha \geq 6\frac{\max_i |C_i|}{\min_i |C_i|}$. If the size of the sample $n = \Theta(\frac{1}{\rho^2\eta^2}\ln\frac{Nk}{\delta})$, then with probability at least $1 - \delta$, the minimum min-sum cost pruning of the tree in Algorithm 4 is $\mathcal{C} \cap P$.*

Proof. Since the tree is laminar to $\mathcal{C} \cap P$, we know that $\mathcal{C} \cap P$ is a pruning of the tree, and any other pruning can be obtained by splitting some clusters in $\mathcal{C} \cap P$ and joining some others into unions. Intuitively, the clusters in $\mathcal{C} \cap P$ are far apart, so the cost increased by joining different clusters is larger than the cost saved by splitting clusters. This claim then implies $\mathcal{C} \cap P$ is the minimum cost pruning. We first prove a

similar claim for $\mathcal{C}$ by the α-perturbation resilience, i.e. for any three different clusters $C_i, C_j, C_l \in \mathcal{C}$, any $A_X \subseteq C_i$, $\alpha d(A_X, C_i \setminus A_X) < d(C_j, C_l)$. Then we prove the claim for $\mathcal{C} \cap P$: for any $A \subseteq C_i \cap P, d(A, C_i \cap P \setminus A) < d(C_j \cap P, C_l \cap P)/2$. Finally we use it to prove $\mathcal{C} \cap P$ is the minimum cost pruning.

First, for any $A_X \subseteq C_i$, we define a perturbation as follows: blow up the distances between the points in A_X and those in $C_i \setminus A_X$ by a factor of α, and keep all the other pairwise distances unchanged. By the α-perturbation resilience, we know that $\mathcal{C}$ is still the optimum clustering after perturbation. Therefore, it has lower cost than the clustering obtained by replacing C_i with A_X and $C_i \setminus A_X$, and replacing C_j and C_l with $C_j \cup C_l$. After canceling the common terms in the costs of the two clusterings, we have $2d'(A_X, C_i \setminus A_X) < 2d'(C_j, C_l)$, which leads to

$$\alpha d(A_X, C_i \setminus A_X) < d(C_j, C_l).$$

Second, we prove the following claim: for any $A \subseteq C_i \cap P$,

$$2d(A, C_i \cap P \setminus A) \leq 2d(C_i \cap P, C_i \cap P) < d(C_j \cap P, C_l \cap P).$$

On one hand, by summing over all the subsets of C_i, we have $\sum_{A_X \subseteq C_i} d(A_X, C_i \setminus A_X) = 2^{|C_i|} d(C_i, C_i)/2$. Then $\frac{\alpha}{2} d(C_i, C_i) < d(C_j, C_l)$. On the other hand, similar to the proof of Lemma 8, we can show that with high probability, for any $p \in C_i, d(p, C_i \cap P) \leq (1+\upsilon)\frac{n}{N} d(p, C_i)$ for $\upsilon = 1/20$. So we have

$$\begin{aligned} d(C_i \cap P, C_i \cap P) &= \sum_{p \in C_i \cap P} d(p, C_i \cap P) \leq (1+\upsilon)\frac{n}{N} \sum_{p \in C_i \cap P} d(p, C_i) \\ &= (1+\upsilon)\frac{n}{N} \sum_{q \in C_i} d(C_i \cap P, q) \leq (1+\upsilon)^2 \frac{n^2}{N^2} d(C_i, C_i). \end{aligned}$$

A similar argument shows that for any C_j and C_l, $d(C_j \cap P, C_l \cap P) \geq (1-\upsilon)^2 \frac{n^2}{N^2} d(C_j, C_l)$. The claim then follows by combining the three inequalities and noting $\upsilon = 1/20$.

Now, we use the claim to prove the optimality of $\mathcal{C} \cap P$. Suppose a pruning $\mathcal{P}^*$ is obtained by splitting h clusters in $\mathcal{C} \cap P$ and at the same time joining some

other clusters into g unions. Specifically, for $1 \le i \le h$, split $C_i \cap P$ into $m_i \ge 2$ clusters $P_{i,1}, \ldots, P_{i,m_i}$; after that, merge $C_{h+1} \cap P, \ldots, C_{h+l_g} \cap P$ into g unions, i.e. for $1 \le j \le g$, $l_0 = 0$, merge $l_j - l_{j-1} \ge 2$ clusters $C_{h+l_{j-1}+1} \cap P, \ldots, C_{h+l_j} \cap P$ into a union U_j; the other clusters in $\mathcal{C} \cap P$ remain the same in $\mathcal{P}^*$. Since the number of clusters is still k, we have $\sum_i m_i - h = l_g - g$. The cost saved by splitting h clusters is

$$\sum_{1 \le i \le h} \sum_{1 \le p \ne q \le m_i} d(P_{i,p}, P_{i,q}) = \sum_{1 \le i \le h} \sum_{1 \le p \le m_i} d(P_{i,p}, C_i \cap P \setminus P_{i,p}). \tag{7}$$

The cost increased by joining clusters is

$$\sum_{1 \le j \le g} \sum_{h+l_{j-1} < p \ne q \le h+l_j} d(C_p \cap P, C_q \cap P). \tag{8}$$

To prove $\mathcal{C} \cap P$ is the minimum cost pruning, we need to show that the saved cost (7) is less than the increased cost (8). Since each term in (8) is twice larger than any term in (7), it suffices to show that the number of the terms in (8) is at least half the number of the terms in (7). Formally, we need to show $2\sum_{1 \le j \le g} \binom{l_j - l_{j-1}}{2} \ge \sum_{1 \le i \le h} m_i$. We have $2\sum_j \binom{l_j - l_{j-1}}{2} = \sum_j (l_j - l_{j-1})(l_j - l_{j-1} - 1) \ge 2\sum_j (l_j - l_{j-1} - 1) = 2(l_g - g)$, where the inequality comes from $l_j - l_{j-1} \ge 2$. Since $l_g - g = \sum_i m_i - h$, it is sufficient to show $l_g - g \ge h$. This comes from $l_g - g = \sum_i m_i - h = \sum_i (m_i - 1) \ge \sum_i 1 = h$ since $m_i \ge 2$. □

2.5 (α, ϵ)-Perturbation Resilience for the Min-Sum Objective

For (α, ϵ)-perturbation resilient min-sum instances, we will show that when $\alpha = \Omega(\frac{\max_i |C_i|}{\min_i |C_i|}), \epsilon = \tilde{O}(\frac{\min_i |C_i|}{n})$, there exists a polynomial time algorithm that outputs a clustering that is both a good approximation and also $\tilde{O}(\epsilon)$-close to the optimal clustering. Formally,

Theorem 8. *Suppose the instance is (α, ϵ)-perturbation resilient to the min-sum objective for $\alpha > \frac{8 \max_i |C_i|}{\min_i |C_i|}$, $\epsilon < \frac{\min_i |C_i|}{600n \log n}$. There exists an algorithm that outputs a*

clustering which is a $(1+\frac{40\epsilon n \log n}{\min_i |C_i|})$*-approximation to the optimal clustering in polynomial time. Furthermore, the output clustering is also* $(6\epsilon \log n)$*-close to the optimal clustering.*

Since $\epsilon = O(\frac{\min_i |C_i|}{n \log n})$, the approximation factor is always $O(1)$ and gets better if ϵ gets smaller. To prove the theorem, we first derive new useful structural properties implied by (α, ϵ)-perturbation resilience for min-sum, and then use them to design our algorithm achieving the guarantees in the theorem.

2.5.1 Structure of (α, ϵ)-Perturbation Resilient Instances

In this subsection, we introduce the notion of bad points for the min-sum objective, and then show that there are just a few bad points while the other points have useful properties. More precisely,

Definition 7. *Define bad points to be those that are not* β *times closer to its own cluster than to other clusters, where* $\beta = \min\left\{\frac{4}{5}\alpha, 8n\right\}$*. That is*

$$B_i = \{p \in C_i : \exists j, d(p, C_j) \leq \beta d(p, C_i)\}, \beta = \min\left\{\frac{4}{5}\alpha, 8n\right\}, B = \cup_i B_i. \quad (9)$$

The other points are called good points.

β is chosen to be $\min\left\{\frac{4}{5}\alpha, 8n\right\}$ for reasons that will become clear in the proof bounding the number of bad points. Informally, we will show that when α is sufficiently large and ϵ is sufficiently small, the number of bad points are bounded by $\tilde{O}(\epsilon n)$ (Theorem 9 in Section 2.5.1.1). The good points in different optimal clusters, by definition, are far from each other. It is then possible to design approximation algorithms if the influence of the few bad points can be eliminated.

However, we do not know the actual bad points. A key observation is that we can introduce a proxy called potential bad points, which can be easily computed. Formally,

Definition 8. *(a) Define* $m_B := 6\epsilon \log n$.

(b) For a set A *with* $|A| > 2m_B$*, define the potential bad points* $F(A)$ *to be the* $2m_B$ *points in* A *that are farthest from* A*. That is,* $F(A) \subseteq A, |F(A)| = 2m_B$*, and for any* $p \in F(A)$*,* $q \in P \setminus F(A)$*,* $d(p, A) \geq d(q, A)$*. The other points* $P(A) = A \setminus F(A)$ *are called potential good points.*

(c) For a cluster A*, define its robust min-sum cost as* $d_{\text{rs}}(A) := d(P(A), P(A))$*, where* $P(A)$ *are the potential good points in* A*. For a clustering* $\mathcal{C}$*, define its robust min-sum cost as* $\sum_{C \in \mathcal{C}} d_{\text{rs}}(C)$.

We show that the robust min-sum cost computed after removing the potential bad points approximates the min-sum cost computed after removing the actual bad points (see Section 2.5.1.2). In other words, we can design approximation algorithms using the potential bad points as if we knew the actual bad points.

2.5.1.1 Bounding the Number of Bad Points

Here we bound the number of bad points by $\tilde{O}(\epsilon n)$ when α is sufficiently large and ϵ is sufficiently small.

Theorem 9. *Suppose the clustering instance is* (α, ϵ)*-perturbation resilient for the min-sum objective where* $\alpha > 4$ *and* $\epsilon < \frac{\min_i |C_i|}{200n}$*. Then we have* $|B| \leq m_B = 6\epsilon n \log n$.

Proof. Assume for contradiction $|B| > 2\eta\epsilon n$ where $\eta = \left\lceil \log \frac{\max_i \max_{p \in B_i} d(p, C_i)}{\min_i \min_{p \in B_i} d(p, C_i)} \right\rceil$. We will first construct a perturbation which leads to a contradiction, and then show that $\eta \leq 3 \log n$, completing the proof.

We begin constructing the perturbation by introducing some notations. Consider the η intervals as follows: $[2^{t-1}v, 2^t v]$ where $v = \min_i \min_{p \in B_i} d(p, C_i), 1 \leq t \leq \eta$. At least one of the intervals, say $[r, 2r]$, will contain the costs of more than $2\epsilon n$ bad points. Let $\hat{B}$ denote an arbitrary subset of $2\epsilon n$ bad points in this interval. Let $\hat{B}_i = \hat{B} \cap C_i$ denote the selected bad points in the optimal cluster C_i. Let $K_i = C_i \setminus \hat{B}_i$ denote the other points in C_i, and set $K = \cup_i K_i$. Denote as D_j all those selected bad points

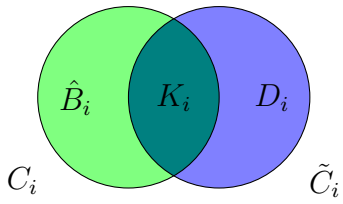


Figure 5: Perturbation construction.

whose second nearest cluster is C_j, that is, $D_j = \{p : p \in \hat{B}_i, j = \arg\min_{\ell \neq i} d(p, C_\ell)\}$. Finally, let $\tilde{C}_i = K_i \cup D_i$. See Figure 5 for an illustration.

Now we are ready to construct the perturbation, which tries to make the selected bad points move to their second nearest clusters and keep the other points in their original clusters. More precisely, the perturbation is constructed as follows: blow up all distances by a factor of α except those within $\tilde{C}_i, 1 \leq i \leq k$. Intuitively, this perturbation favors the clustering $\tilde{C}_i$.

To derive a contradiction, consider the optimal clustering after perturbation, denoted as $\{C_i'\}$. Since there are more than ϵn bad points in $\hat{B}$, by (α, ϵ)-perturbation resilience, not all of them move to new clusters in $\{C_i'\}$, and thus $\{C_i'\}$ is different from $\{\tilde{C}_i\}$. In fact, we will show that the clustering $\{\tilde{C}_i\}$ has a lower cost than $\{C_i'\}$, which is a contradiction. To do so, we consider changing $\{C_i'\}$ to $\{\tilde{C}_i\}$ by moving points. It is sufficient to show that by moving these points, the cost saved is larger than the cost added.

To bound the costs, we first divide the points into different types. See Figure 6 for an illustration. First, we need to move out $C_i' \setminus \tilde{C}_i$ from each C_i'. These points can be divided into three types:

(1) $U_i = C_i' \cap \hat{B}_i$ are the selected bad points in C_i that need to be moved out.

(2) $V_i = (C_i' \setminus \tilde{C}_i) \cap (\cup_{j \neq i} \hat{B}_j) = \cup_{j \neq i} (\hat{B}_j \cap C_i')$ are the selected bad points that are

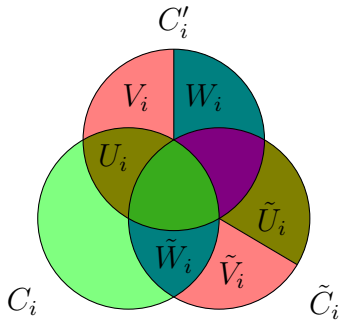


Figure 6: Different types of points when bounding the number of bad points for (α, ϵ)-perturbation resilient min-sum instances.

from other optimal clusters. But their second nearest cluster is not C_i, so they are not in $\tilde{C}_i$.

(3) $W_i = (C'_i \setminus \tilde{C}_i) \cap (\cup_{j \neq i} K_j) = \cup_{j \neq i}(K_j \cap C'_i)$ are points that are from K_j for some $j \neq i$ and are in C'_i. But they are not from K_i and thus are not in $\tilde{C}_i$.

Second, we need to move in $\tilde{C}_i \setminus C'_i$ for each $\tilde{C}_i$. Similarly, these points can also be divided into three types:

(1) $\tilde{W}_i = K_i \setminus C'_i = \cup_{j \neq i}(K_i \cap C'_j)$ are those points in K_i and in C'_j for some $j \neq i$. This means that they are points in $\cup_j W_j$. More specifically, we have $\cup_i \tilde{W}_i = \cup_j W_j$.

(2) $\tilde{V}_i = (D_i \setminus C'_i) \cap \left[\cup_{\ell \neq j}(\hat{B}_\ell \cap C'_j)\right]$ are part of the selected bad points whose second nearest cluster is C_i. They are originally in $\hat{B}_\ell$ for some ℓ but are in C'_j for some $j \neq \ell$. In other words, they are points from V_j for some j, and we have $\cup_i \tilde{V}_i = \cup_j V_j$.

(3) $\tilde{U}_i = (D_i \setminus C'_i) \cap \left[\cup_j(\hat{B}_j \cap C'_j)\right]$ are also part of the selected bad points whose second nearest cluster is C_i. They are originally in $\hat{B}_j$ for some j and are also in C'_j. In other words, they are points from U_j for some j, and we have $\cup_i \tilde{U}_i = \cup_j U_j$.

In conclusion, we have $C_i' = C_i' \cap \tilde{C}_i + U_i + V_i + W_i$, $\tilde{C}_i = C_i' \cap \tilde{C}_i + \tilde{U}_i + \tilde{V}_i + \tilde{W}_i$. We also have $\cup_i \tilde{U}_i = \cup_j U_j$, $\cup_i \tilde{V}_i = \cup_j V_j$, and $\cup_i \tilde{W}_i = \cup_j W_j$. The costs saved and added are then summarized as follows. Suppose we first move out W, then V, and finally U. The cost saved by moving out W is at least $2\sum_i d'(W_i, C_i' \cap C_i)$, that saved by moving out V is at least $2\sum_i d'(V_i, C_i' \cap C_i)$, and that saved by moving out U is at least $2\sum_i d'(U_i, C_i' \cap K_i)$. Next, we move in $\tilde{W}$, then $\tilde{V}$, and finally $\tilde{U}$. The cost added by moving in $\tilde{W}$ is at most $2\sum_i d'(\tilde{W}_i, \tilde{W}_i + C_i' \cap \tilde{C}_i)$, that added by moving in $\tilde{V}$ is at most $2\sum_i d'(\tilde{V}_i, \tilde{C}_i)$, and that added by moving in $\tilde{U}$ is at most $2\sum_i d'(\tilde{U}_i, \tilde{C}_i)$.

We are now ready to show that the cost saved is greater than the cost added. The high level idea is that a significant amount of cost is saved by moving U_i to the correct clusters, while the costs added by moving V_i and W_i are generally small since the number of points moved is bounded by $3\epsilon n$ and the cost of the selected bad points moved is bounded by $2r$. Formally, we have the following claim, whose proof is presented in Appendix A.4.1.

Claim 3. *The costs saved and added by moving $\{U_i\}_{i=1}^k$, $\{V_i\}_{i=1}^k$ and $\{W_i\}_{i=1}^k$ satisfy:*

$$
\begin{aligned}
(a) \qquad & 2\sum_i d'(U_i, C_i' \cap K_i) - 2\sum_i d'(\tilde{U}_i, \tilde{C}_i) \\
\geq \; & \frac{3}{10}\alpha \sum_i d(U_i, C_i) - \frac{2\alpha}{100}\sum_i d(W_i, C_i) - \frac{8\alpha + 16}{100} r\epsilon n, \\
(b) \qquad & 2\sum_i d'(V_i, C_i' \cap C_i) - 2\sum_i d'(\tilde{V}_i, \tilde{C}_i) \\
\geq \; & \frac{99}{50}(\alpha - 2)\sum_i d(V_i, C_i) - \frac{2\alpha}{100}\sum_i d(W_i, C_i) - \frac{8\alpha + 16\beta}{100} r\epsilon n, \\
(c) \qquad & 2\sum_i d'(W_i, C_i' \cap C_i) - 2\sum_i d'(\tilde{W}_i, \tilde{W}_i + C_i' \cap \tilde{C}_i) \\
\geq \; & \frac{98}{50}(\alpha - 2)\sum_i d(W_i, C_i) - \frac{8\alpha + 8\beta}{100} r\epsilon n.
\end{aligned}
$$

After adding up all the inequalities in the claim, the right hand side is a lower bound on the difference between the cost saved and the cost added, which we now show must be positive when $\alpha > 4$ and $\beta \leq \frac{4}{5}\alpha$. The terms about $d(W_i, C_i)$ and

$d(V_i, C_i)$ are positive, so it suffices to show that $\sum_i d(U_i, C_i)$ is larger than $r\epsilon n$. First, $d(p, C_i) \geq r$ for any $p \in U_i$. Second, $|\cup_i U_i| \geq \epsilon n$ since there are $2\epsilon n$ selected bad points but no more than ϵn of them move from their original clusters in $\{C_i\}$ to a different cluster in $\{C'_i\}$. Then we have $\sum_i d(U_i, C_i) \geq \sum_i r|U_i| = r\sum_i |U_i| \geq r\epsilon n$. Hence, the difference between the cost saved and the cost added is positive. This means the cost of $\{\tilde{C}_i\}$ is smaller than the cost of $\{C'_i\}$, which contradicts the assumption that $\{C'_i\}$ is the optimal clustering under d'. Therefore, there can be at most $2\eta\epsilon n$ bad points.

Finally, what is left is to show $\eta \leq 3\log n$. Suppose p_1 is the point that achieves $\max_i \max_{p\in B_i} d(p, C_i)$ and p_2 is the point that achieves $\min_i \min_{p\in B_i} d(p, C_i)$. Without loss of generality, suppose $p_1 \in C_1$ and $p_2 \in C_2$. By definition of bad points, there exists $C_i \neq C_2$ such that $d(p_2, C_i) \leq \beta d(p_2, C_2)$. If $C_i \neq C_1$, we have $d(p_1, C_1) \leq d(C_2, C_i)$, since otherwise we can get lower cost by splitting C_1 into p_1 and $C_1 \setminus \{p_1\}$ while merging C_2 and C_i. If $C_i = C_1$, we also have $d(p_1, C_1) \leq d(C_2, C_i)$, since otherwise we can get lower cost by splitting C_1 into p_1 and $C_1 \setminus \{p_1\}$ and then merging C_2 and $C_1 \setminus \{p_1\}$. In both cases, we have

$$\begin{aligned} d(p_1, C_1) \leq d(C_2, C_i) &\leq |C_i| d(p_2, C_2) + |C_2| d(p_2, C_i) \\ &\leq |C_i| d(p_2, C_2) + \beta |C_2| d(p_2, C_2) \\ &\leq 8n^2 d(p_2, C_2) \end{aligned}$$

where the last inequality follows from $\beta \leq 8n$. Then we have $\eta \leq 3\log n$. □

2.5.1.2 Properties of Actual and Potential Good Points

Since there are just a few bad points and the good points in different clusters are far apart, the cost between sufficiently large subsets of their good points accounts for most of the cost between the two clusters. This means that the min-sum cost can be approximately computed on the good points, and the min-sum clustering can be approximately solved if we knew the actual good points.

Lemma 10. *Suppose $W_i \subseteq G_i, W_j \subseteq G_j$. When $|C_i| \geq 50|C_i \setminus W_i|$ and $|C_j| \geq 50|C_j \setminus W_j|$, we have $d(C_i, C_j) \leq \frac{3}{2} d(W_i, W_j)$.*

Proof Sketch. Since $|W_i| \approx |C_i|$ and $|W_j| \approx |C_j|$, it suffices to show that $d_{\mathrm{a}}(C_i, C_j) \leq d_{\mathrm{a}}(W_i, W_j)$ approximately. By the triangle inequality, $d_{\mathrm{a}}(C_i, C_j) \leq d_{\mathrm{a}}(C_i, W_i) + d_{\mathrm{a}}(W_i, W_j) + d_{\mathrm{a}}(W_j, C_j)$, so we only need to bound $d_{\mathrm{a}}(W_i, C_i)$ and $d_{\mathrm{a}}(W_j, C_j)$ by $d_{\mathrm{a}}(W_i, W_j)$.

By definition of good points, $d_{\mathrm{a}}(W_i, C_i)$ is much less than $d_{\mathrm{a}}(W_i, C_j)$, which is approximately $d_{\mathrm{a}}(W_i, W_j)$ since W_j takes up a majority of points in C_j. A similar argument bounds $d_{\mathrm{a}}(W_j, C_j)$, which then leads to the lemma. The complete proof is provided in Appendix A.4. □

Furthermore, the good points in different optimal clusters are far apart in the following sense: the good points from two different clusters have cost much larger than those in a third cluster have. Formally,

Lemma 11. *For any three different optimal clusters C_i, C_j, and C_l, and any $A \subset G_i$, $\frac{18}{5} d(A, G_i \setminus A) < d(G_j, G_l)$. Consequently, $\frac{9}{5} d(G_i, G_i) < d(G_j, G_l)$.*

If we can construct a tree on the good points with a pruning that assigns all good points correctly, then this property is useful in finding the pruning.

Now we turn to analyze the potential good points. A key property of the potential good points is the following: for any point p and any sufficiently large set A, the cost between p and the potential good points in A is roughly bounded by the cost between p and any sufficiently large subset H of A. See Lemma 12 for details. A specific case is when H is the actual good points in A. In this case, the property says that the cost between p and the potential good points is roughly bounded by the cost between p and the actual good points. This means that in suitable situations, we can regard potential good points as actual good points.

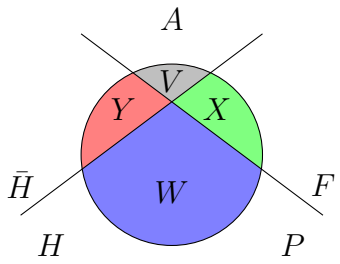


Figure 7: Properties of the potential good points.

Lemma 12. *Suppose $H \subseteq A$ such that $|A \setminus H| \leq m_B$. Let $F = F(A), P = P(A), \bar{H} = A \setminus H$. Let $W = H \cap P, V = \bar{H} \cap F, X = F \cap H, Y = \bar{H} \cap P$. See Figure 7 for an illustration. If $|A| \geq 20m_B$, then for any p, $d(p, P) \leq \frac{|W|+|Y|}{|W|-|X|} d(p, H)$.*

Proof. Since $d(p, P) = d(p, Y) + d(p, W)$ and $d(p, H) = d(p, X) + d(p, W)$, the lemma is true if $Y = \emptyset$. Otherwise, we need to compare $d(p, X)$ and $d(p, Y)$. By the triangle inequality, we have $d_{\mathrm{a}}(W, X) \leq d_{\mathrm{a}}(W, p) + d_{\mathrm{a}}(p, X)$ and $d_{\mathrm{a}}(p, Y) \leq d_{\mathrm{a}}(p, W) + d_{\mathrm{a}}(W, Y)$. Then

$$d(p, X) \geq \frac{d(W, X)}{|W|} - \frac{|X|}{|W|} d(p, W), \qquad d(p, Y) \leq \frac{d(W, Y)}{|W|} + \frac{|Y|}{|W|} d(p, W).$$

From these bounds on $d(p, X)$ and $d(p, Y)$, we have

$$d(p, H) \geq \frac{d(W, X)}{|W|} + \frac{|W| - |X|}{|W|} d(p, W), \qquad d(p, P) \leq \frac{d(W, Y)}{|W|} + \frac{|W| + |Y|}{|W|} d(p, W).$$

The lemma then follows from these two inequalities and the following claim.

Claim 4. $d(X, W) \geq d(Y, W + Y)$.

Intuitively, the points in $X \subseteq F$ are among those farthest away from A, so $d_{\mathrm{a}}(X, A) \geq d_{\mathrm{a}}(Y, A)$. Since $|A \setminus H| \leq m_B$ and $|F| = 2m_B$, we have $|X| \geq 2|Y|$. Then the cost of Y cannot be too large compared to that of X. The complete proof of the claim is provided in Appendix A.4. □

2.5.2 Approximating the Optimal Clustering

In this subsection, we design an approximation algorithm and prove our final result Theorem 8 by utilizing the properties of the (α, ϵ)-perturbation resilience. Note that we can generate a list of sufficiently large almost "pure" blobs using Algorithm 3. However, unlike for (α, ϵ)-perturbation resilient k-median instances, it is not guaranteed that the robust linkage procedure in [16] can link these blobs into a tree so that a pruning of the tree assigns all but bad points correctly. In Section 2.5.2.1, we design a robust average linkage algorithm to achieve this goal, and in Section 2.5.2.2, we show that this pruning can be found in polynomial time. Although this pruning may not be a good approximation to the optimum, in Section 2.5.2.3 we show that a good approximation can be computed by reassigning points in it, which leads to Theorem 8.

2.5.2.1 Constructing A Tree with A Pruning Close to the Optimal Clustering

Here we show that by utilizing the bound on the number of bad points, we can construct a tree with a pruning that assigns all good points correctly. As described in Algorithm 6, we first use Algorithm 3 to generate a list of blobs, and then use a robust version of average linkage to link them into a tree: repeatedly merge the two blobs with the minimum robust average distance defined as follows.

Definition 9. *The robust average distance $d_{ra}(A_1, A_2)$ between two sets A_1, A_2 is defined as the average distance between their potential good points. That is, $d_{ra}(A_1, A_2) = \frac{d(P(A_1), P(A_2))}{|P(A_1)||P(A_2)|}$.*

We now prove that the tree output by Algorithm 6 has a pruning that correctly assigns all good points.

Lemma 13. *The tree output in Algorithm 6 has a pruning $\mathcal{C}'$ that assigns all good points correctly.*

Algorithm 6 Robust Average Linkage

Input: Data set P, distance function $d(\cdot,\cdot)$ on P, $\min_i |C_i|$, $\epsilon > 0$.

1: Use Algorithm 3 with $m_B = 6\epsilon n \log n$ and $\gamma = \frac{4}{5}$ to get a list $\mathcal{L}_0$ of blobs.
2: Initialize the clustering $\mathcal{L}$ with each blob being a cluster.
3: Repeat till only one cluster remains:
merge clusters C, C' which minimize $d_{ra}(C, C')$.
4: Let $\mathcal{T}$ be the tree with blobs as leaves and internal nodes corresponding to the merges performed.

Output: The tree $\mathcal{T}$.

Proof. To analyze the algorithm, we begin with the following property of good points. When combined with the property of Algorithm 3 (Lemma 4), it immediately shows that each blob in the list $\mathcal{L}_0$ has size at least $\frac{1}{2}\min_i |C_i|$, and contains good points from only one optimal cluster.

Claim 5. *For any $p \in G_i$, all its $\frac{4|C_i|}{5}$ nearest neighbors belong to $C_i \cup B$.*

Proof. We need to show that for any $j \neq i$ and any good point $q \in G_j$, $d(p,q)$ is large compared to $d_{\mathrm{a}}(p, C_i)$. Intuitively, p is much farther away from C_j than from C_i: $\beta d(p, C_i) \leq d(p, C_j)$. It suffices to bound $d(p, C_j)$ by $d(p,q)$ and $d(p, C_i)$. By the triangle inequality, we have

$$d(p, C_j) \leq |C_j| d(p,q) + d(q, C_j) \quad \text{and} \quad d(q, C_j) \leq \frac{1}{\beta} d(q, C_i) \leq \frac{|C_i|}{\beta} d(p,q) + \frac{1}{\beta} d(p, C_i).$$

Combining these inequalities, we have $(\beta - \frac{1}{\beta}) d(p, C_i) \leq (|C_j| + \frac{|C_i|}{\beta}) d(p,q)$. When $\alpha > 8\frac{\max_i |C_i|}{\min_i |C_i|}$, we have $5 d_{\mathrm{a}}(p, C_i) < d(p,q)$, which then leads to the conclusion. □

It now suffices to prove by induction that the clustering $\mathcal{L} \cap G$ is always laminar to $\mathcal{C} \cap G$. It is true at the beginning by the property of Algorithm 3. Assume for contradiction that the laminarity is first violated after merging A and D. There are two cases: (1) A and D are strict subsets of different optimal clusters; (2) A is a strict subset of G_i while D is the union of the good points in several optimal clusters. We have the following claims for the two cases respectively.

Claim 6. *(a) Suppose $A \in \mathcal{L}, A \cap G \subsetneq G_i$, and $D \in \mathcal{L}, D \cap G \subsetneq G_j (j \neq i)$. Then there exists $A' \neq A$ in $\mathcal{L}$ such that $A' \cap G \subsetneq G_i$ and $d_{ra}(A, A') < d_{ra}(A, D)$.*
(b) Suppose $A \in \mathcal{L}, A \cap G \subsetneq G_i$, and $D \in \mathcal{L}$, $D \cap G$ is the union of good points in several optimal clusters. Then there exists $A' \neq A$ in $\mathcal{L}$ such that $A' \cap G \subsetneq G_i$ and $d_{ra}(A, A') < d_{ra}(A, D)$.

The intuition is that since good points in different optimal clusters are far away, we can find a blob A' from $C_i \cup B$ such that good points in A' have smaller average distance to good points in A than those in D have, that is, $d_{\mathrm{a}}(A \cap G, A' \cap G) < \frac{1}{2} d_{\mathrm{a}}(A \cap G, D \cap G)$. Then we show that the bad points in these blobs do not change things much: $d_{ra}(A, A')$ is approximately less than $d_{\mathrm{a}}(A \cap G, A' \cap G)$, and $d_{ra}(A, D)$ is approximately greater than $d_{\mathrm{a}}(A \cap G, D \cap G)$. These then lead to the claims. Their complete proofs are presented in Appendix A.5.1.

By these two claims, we should first merge A with A' rather than with D, which is contradictory. So the laminarity is always preserved, which completes the proof. □

2.5.2.2 Getting A Pruning Close to the Optimal Clustering

We have shown that a tree can be constructed such that there is a pruning, denoted as $\mathcal{C}'$, that assigns all good points correctly. Here we show how to find this pruning from the tree. Suppose we can remove the actual bad points and compute the cost between the good points. Since the good points from different clusters are far apart, the good point cost increased by joining different clusters in $\mathcal{C}'$ is larger than that saved by splitting clusters in $\mathcal{C}'$ (Lemma 11). Then any other pruning has larger cost than $\mathcal{C}'$. Unfortunately, we do not know the actual good points. Therefore, we consider the potential good points and compute the robust min-sum cost. It turns out that $\mathcal{C}'$ indeed is the pruning with the minimum robust min-sum cost.

Lemma 14. *Suppose the pruning $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ in tree $\mathcal{T}$ assigns all good points correctly. Then $\mathcal{C}'$ is the minimum robust min-sum cost pruning in the tree.*

Proof Sketch. Computing the robust min-sum cost will eliminate the effect of the bad points and work as if we knew the actual good points: the robust min-sum cost saved by splitting a node is at most the good point cost saved (Claim 7), and the robust min-sum cost increased by merging two nodes is in the same order of the good point cost increased (Claim 8).

Claim 7. *If* $|C_i'| \geq 20m_B$*, then* $d_{\text{rs}}(C_i') \leq d(G_i, G_i)$.

Proof. The claim follows from Claim 4 (See Figure 8 for an illustration of the notations) by setting $A = C_i'$ and $H = G_i$. In particular, we have $d_{\text{rs}}(A) = d(P, P) \leq d(W, W) + 2d(Y, W + Y)$ and $d(H, H) \geq d(W, W) + 2d(X, W)$. By Claim 4, $d(Y, W + Y) \leq d(X, W)$, which completes the proof. □

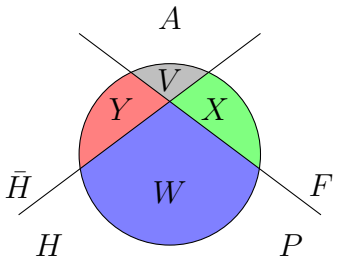


Figure 8: Notations in Claim 4 and 7.

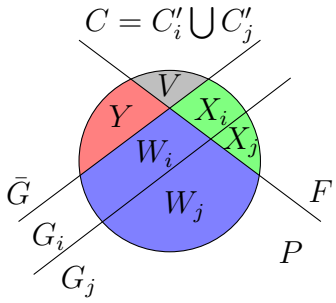


Figure 9: Notations in Claim 8.

Claim 8. *For* $t \in \{i, j\}$, $|C_t| \geq 100m_B$*, and* C_t' *contains all good points in* C_t *but no good points in other optimal clusters. Then* $d_{\text{rs}}(C_i' \cup C_j') - d(G_i, G_i) - d(G_j, G_j) \geq (\frac{4}{3} - \frac{4}{\beta})d(G_i, G_j)$.

Proof. Let $C = C_i' + C_j', F = F(C), P = P(C)$, and $G = G_i + G_j, \bar{G} = C \setminus G$. Define $W_i = G_i \cap P, X_i = G_i \cap F$; define W_j, X_j similarly. Also, define $Y = P \cap \bar{G}, V = F \cap \bar{G}$. See Figure 9. Then

$$d(C \setminus F, C \setminus F) - d(G_i, G_i) - d(G_j, G_j) \geq 2d(W_i, W_j) - 2d(X_i, G_i) - 2d(X_j, G_j). \quad (10)$$

We have $d(X_i, C_i) + d(X_j, C_j) \leq \frac{d(X_i,C_j)+d(X_j,C_i)}{\beta} \leq \frac{2}{\beta} d(C_i, C_j)$ by the definition of good points, and $d(C_i, C_j) \leq \frac{3}{2} d(W_i, W_j)$ by Lemma 10. So (10) is at least $(\frac{4}{3} - \frac{4}{\beta}) d(C_i, C_j) \geq (\frac{4}{3} - \frac{4}{\beta}) d(G_i, G_j)$. □

By these claims and Lemma 11, when splitting some node in the pruning while merging some other two, the cost increased is larger than the cost saved. Since any pruning of size k can be obtained from $\mathcal{C}'$ by splitting some nodes while merging some others, we can show that $\mathcal{C}'$ has the minimum robust min-sum cost. See Appendix A.5.2 for the complete proof. □

2.5.2.3 Getting a Constant Factor Approximation

We have showed that the pruning $\mathcal{C}'$ that assigns all good points correctly has minimum robust min-sum cost, so we can use dynamic programming on the tree to get the pruning. However, this pruning may not be a good approximation. For example, consider an instance consisting of two unbalanced clusters. Assume that there is only one bad point, belonging to the small cluster. Further assume the distances between the good points in each cluster are negligible, then assigning the bad point incorrectly to the large cluster will lead to an $O(\frac{\max_i |C_i|}{\min_i |C_i|})$-approximation. So the pruning $\mathcal{C}'$ may not be a constant approximation.

Algorithm 7 Getting a constant factor approximation

Input: A clustering $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$, where $G_i \subseteq C'_i \subseteq C_i \cup B$.

1: **for** each point p **do**
2: Assign p to the index i such that $d(p, P(C'_i))$ is minimized.
3: **end for**
4: Let C''_i be the set of points assigned to the index i.

Output: The clustering $\mathcal{C}'' = \{C''_1, \ldots, C''_k\}$.

Here we show that a constant factor approximation can be computed by reassigning the points in $\mathcal{C}'$. As described in Algorithm 7, we reassign each point p to the index i that minimizes the cost between p and the potential good points in the cluster

C'_i. To analyze Algorithm 7, we first prove that after reassignment all good points are still assigned correctly (Lemma 15), and then bound the cost.

Lemma 15. *For any $p \in G_i$, any $j \neq i$, $d(p, P(C'_j)) > d(p, P(C'_i))$.*

Proof Sketch. By Lemma 12, $d(p, P(C'_i)) \approx d(p, C_i)$. By definition of good points, $\beta d(p, C_i) \leq d(p, C_j)$, so it suffices to show that $d(p, P(C'_j))$ is not so small compared to $d(p, C_j)$. Let $W_j = G_j \cap P(C'_j)$ denote the good points that are also potential good points, and let $Z_j = C_j \setminus W_j$ denote all other points in C_j. Since $W_j \subseteq P(C'_j)$, we only need to prove that $d(p, W_j)$ is large compared to $d(p, Z_j)$. Intuitively, this is true since p is far from good points in G_j, and W_j contains most of G_j. See Appendix A.5.3 for details. □

Theorem 8. *Suppose the instance is (α, ϵ)-perturbation resilient to the min-sum objective for $\alpha > \frac{8 \max_i |C_i|}{\min_i |C_i|}$, $\epsilon < \frac{\min_i |C_i|}{600n \log n}$. There exists an algorithm that outputs a clustering which is a $\left(1 + \frac{40\epsilon n \log n}{\min_i |C_i|}\right)$-approximation to the optimal clustering in polynomial time. Furthermore, the output clustering is also $(6\epsilon \log n)$-close to the optimal clustering.*

Proof. By Lemma 15, all the good points in C_i are assigned correctly to C''_i. Let $A_i = C''_i \setminus G_i$ denote all the bad points assigned to C''_i. The cost of the output clustering $\mathcal{C}''$ can be written as follows.

$$\begin{aligned} \sum_i d(C''_i, C''_i) &= \sum_i d(G_i + A_i, G_i + A_i) \\ &= \sum_i d(G_i, G_i) + 2\sum_i d(G_i, A_i) + \sum_i d(A_i, A_i). \end{aligned} \tag{11}$$

We need to bound the last two terms.

Let $r = \frac{\min_i |C_i|}{m_B}$. By the triangle inequality, we have $d_{\mathrm{a}}(A_i, A_i) \leq 2d_{\mathrm{a}}(A_i, G_i)$, leading to

$$d(A_i, A_i) \leq \frac{2|A_i|}{|G_i|} d(A_i, G_i) \leq \frac{2m_B}{|C_i| - m_B} d(A_i, G_i) \leq \frac{2}{r-5} d(A_i, G_i). \tag{12}$$

So it suffices to bound $d(A_i, G_i)$. In Appendix A.5.3, we prove

Claim 9. $\sum_i d(A_i, G_i) \leq \frac{r^2}{(r-5)^2} \sum_i d(C_i, C_i) - \frac{r^2-1}{(r-5)^2} \sum_i d(G_i, G_i)$.

The intuition is as follows. Suppose $p \in A_i$ comes from C_j. Then $d(p, G_i) \approx d(p, P(C'_i))$, which is smaller than $d(p, P(C'_j))$ since p is assigned to i. By Lemma 12, we have $d(p, P(C'_j)) \approx d(p, G_j)$. Applying this argument for all $p \in A_i$ and all i, and noting that $\cup_i A_i = \cup_j B_j$, we have $\sum_i d(A_i, G_i)$ is approximately less than $\sum_j d(B_j, G_j) \leq \sum_j [d(C_j, C_j) - d(G_j, G_j)]$.

The proof of correctness is completed by combining this claim and the inequalities (11) and (12).

Running Time Algorithm 3 takes time $O(n^{\omega+1})$ (as shown in the proof of Theorem 4), and the rest steps of Algorithm 6 take time $O(n^3)$. Finding the minimum robust min-sum cost pruning in the tree output by Algorithm 6 takes time $O(n^3)$, and Algorithm 7 takes time $O(n^3)$. So the total running time is $O(n^{\omega+1})$. □

CHAPTER III

DISTRIBUTED CLUSTERING

Many modern applications face an explosion of data. Usually, the data is distributed over different locations, such as distributed databases [91, 36], images and videos over networks [87], surveillance [53] and sensor networks [35, 54]. In many of these applications the data are inherently distributed because, as in sensor networks, it is collected at different sites. Since most classic clustering algorithms are designed for the centralized setting, it has become crucial to develop clustering algorithms which are effective in the distributed setting.

Several algorithms for distributed clustering have been proposed and empirically tested. Some of these algorithms [48, 101, 38] are direct adaptations of centralized algorithms which rely on statistics that are easy to compute in a distributed manner. Other algorithms [60, 64] generate summaries of local data and transmit them to a central coordinator which then performs the clustering algorithm. No theoretical guarantees are provided for the clustering quality in these algorithms, and they do not try to minimize the communication cost. Additionally, most of these algorithms assume that the distributed nodes can communicate with all other sites or that there is a central coordinator that communicates with all other sites.

In this chapter, we study the problem of distributed k-median and k-means clustering where the data is distributed across nodes whose communication is restricted to the edges of an arbitrary graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. Our technique for reducing communication in general graphs is based on the construction of a small set of points called coreset, which act as a proxy for the entire data set. For k-means, we further

provide an algorithm for high dimensional data, where we first reduce the dimension of the data by distributed principal component analysis (PCA) and then apply our distributed clustering algorithm to the projected data.

An ϵ-*coreset* is a weighted set of points whose cost on any set of centers is approximately the cost of the original data on those same centers up to accuracy ϵ. Thus an approximate solution for the coreset is also an approximate solution for the original data. Coresets have previously been studied in the centralized setting ([57, 45]) but have also recently been used for distributed clustering as in [104] and as implied by [47]. In Section 3.2, we propose a distributed algorithm for k-means and k-median, by which each node constructs a local portion of a global coreset. The nodes then share the local portions of the coreset, which can be done efficiently in general graphs using a message passing approach. More precisely, each node computes an approximate solution for its local data and communicate the cost of this local solution, and then constructs the local portion of a global coreset using only its local data and the total cost of each node's local solution. For ϵ constant, this builds a coreset of size $\tilde{O}(kd+sk)$ for k-median and k-means when the data lies in d dimensions and is distributed over s sites[1]. If there is a central coordinator among the n sites, then clustering can be performed on the coordinator by collecting the local portions of the coreset with a communication cost equal to the coreset size $\tilde{O}(kd + sk)$. For distributed clustering over general connected topologies, we propose an algorithm based on the distributed coreset construction and a message-passing approach, whose communication cost improves over previous coreset-based algorithms. Experimental results on large scale data sets show that our algorithm performs well in practice. For a fixed amount of communication, our algorithm outperforms other coreset construction algorithms.

[1]For k-median and k-means in general metric spaces, the bound on the size of the coreset can be obtained by replacing d with the logarithm of the total number of points. The analysis for general metric spaces is largely the same as that for d dimensional Euclidean space, so we will focus on Euclidean space and point out the difference when needed.

In the above algorithms, the number of points in the coreset is independent of the number of the original data points, which is useful for large-scale applications. However, it is linear in the dimension of the data, leading to high communication cost for high dimensional data. In Section 3.3, we propose a distributed PCA algorithm, and show that its output represents the original data in the sense that any good approximation solution of k-means clustering on the output projected data is also a good solution on the original data. When combined with the distributed clustering algorithm in Section 3.2, this leads to an algorithm whose communication cost (in terms of the number of points communicated) is independent of the size and the dimension of the original data. Our experiment results demonstrate that this significantly reduces the communication cost while hardly comprising the quality of the k-means clustering solutions.

3.1 Preliminaries

Let $d(p, q)$ denote the Euclidean distance between any two points $p, q \in \mathbf{R}^d$. The goal of k-means clustering is to find a set of k centers $\mathbf{x} = \{x_1, x_2, \ldots, x_k\}$ which minimize the k-means cost of data set $P \subseteq \mathbf{R}^d$. Here the k-means cost is defined as $\text{cost}(P, \mathbf{x}) = \sum_{p \in P} d(p, \mathbf{x})^2$ where $d(p, \mathbf{x}) = \min_{x \in \mathbf{x}} d(p, x)$. If P is a weighted data set with a weighting function w, then the k-means cost is defined as $\sum_{p \in P} w(p) d(p, \mathbf{x})^2$. Similarly, the k-median cost is defined as $\sum_{p \in P} d(p, \mathbf{x})$. Both k-means and k-median cost functions are known to be **NP**-hard to minimize (see for example [7]). For both objectives, there exist several readily available polynomial-time algorithms that achieve constant approximation solutions (see for example [63, 76]).

In the distributed clustering task, we consider a set of s nodes $V = \{v_i, 1 \leq i \leq s\}$ which communicate on an undirected connected graph $G = (V, E)$ with $m = |E|$ edges. More precisely, an edge $(v_i, v_j) \in E$ indicates that v_i and v_j can communicate with each other. Here we measure the communication cost in number of points

transmitted, and assume for simplicity that there is no latency in the communication. On each node v_i, there is a local set of data points P_i, and the global data set is $P = \cup_{i=1}^{s} P_i$. The goal is to find a set of k centers $\mathbf{x}$ which optimize $\text{cost}(P, \mathbf{x})$ while keeping the computation efficient and the communication cost as low as possible. Our focus is to reduce the total communication cost while preserving theoretical guarantees for approximating clustering cost.

3.1.1 Coresets

For the distributed clustering task, a natural approach to avoid broadcasting raw data is to generate a local summary of the relevant information. If each site computes a summary for their own data set and then communicates this to a central coordinator, a solution can be computed from a much smaller amount of data, drastically reducing the communication.

In the centralized setting, the idea of summarization with respect to the clustering task is captured by the concept of coresets [57, 45]. A coreset is a set of points, together with a weight for each point, such that the cost of this weighted set approximates the cost of the original data for any set of k centers. The formal definition of coresets is:

Definition 10 (**coreset**). *An ϵ-coreset for a set of points P with respect to a center-based cost function is a set of points S and a set of weights $w : S \to \mathbf{R}$ such that for any set of centers $\mathbf{x}$,*

$$(1-\epsilon)\text{cost}(P, \mathbf{x}) \leq \sum_{p \in S} w(p)\text{cost}(p, \mathbf{x}) \leq (1+\epsilon)\text{cost}(P, \mathbf{x}).$$

In the centralized setting, many coreset construction algorithms have been proposed for k-median, k-means and some other cost functions. For example, for points in $\mathbf{R}^d$, algorithms in [45] construct coresets of size $t = \tilde{O}(kd/\epsilon^4)$ for k-means and coresets of size $t = \tilde{O}(kd/\epsilon^2)$ for k-median. In the distributed setting, it is natural to ask

whether there exists an algorithm that constructs a small coreset for the entire point set but still has low communication cost. Note that the union of coresets for multiple data sets is a coreset for the union of the data sets. The immediate construction of combining the local coresets from each node would produce a global coreset whose size was larger by a factor of n, greatly increasing the communication complexity. We present a distributed algorithm which constructs a global coreset the same size as the centralized construction and only needs a single value[2] communicated to each node. This serves as the basis for our distributed clustering algorithm.

3.1.2 Principal Component Analysis

PCA is a classical tool for dimension reduction, and has been closely related to k-means [41, 70]. In Section 3.3, we first use PCA on high dimensional data and then do distributed clustering on the projected data, which leads to lower communication cost. We introduce the following notations for PCA. View the local data P_i as a matrix, whose rows are data points. The global data P is then a concatenation of the local data matrix, i.e. $P^\top = [P_1^\top, P_2^\top, \ldots, P_s^\top]$. For simplicity, we always assume the data is centered, that is, $\sum_{p \in P} p = 0$; otherwise, we can first perform a step to center the data, whose communication and computation cost will be dominated by the other steps in our algorithms.

For a matrix $X = [x_{ij}]$, let $\|X\|_F^2 = \sum_{i,j} x_{i,j}^2$. We say that X has orthonormal columns if its columns are orthogonal unit vectors. Let $L(X)$ denote the linear subspace spanned by the columns of X. For simplicity, for a set of points P, we denote $d^2(P, L(X)) = \sum_{p \in P} d(p, L(X))^2$.

For a point p, let $\Pi_X(p)$ denote its projection to $L(X)$. Note that for an orthogonal matrix X, the projection of a point p to $L(X)$ will be pX using the coordinates with

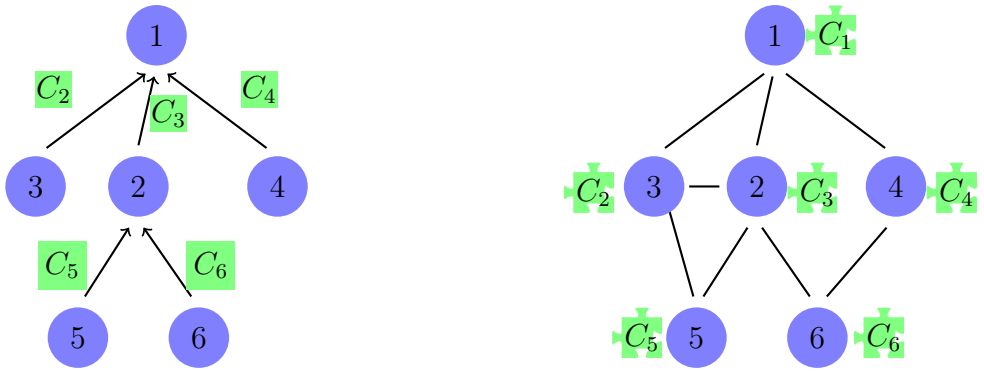
[2]The value that is communicated is the sum of the costs of approximations to the local optimal clustering. This is guaranteed to be no more than a constant factor times larger than the optimal cost.

respect to the column space of X, and will be pXX^T using the original coordinates.

3.2 Distributed Coreset-Based Clustering

Since coresets summarize local information they are a natural tool to use when trying to reduce communication complexity. If each node constructs an ϵ-coreset on its local data, then the union of these coresets is clearly an ϵ-coreset for the entire data set. Unfortunately the size of the coreset in this approach increases greatly with the number of nodes.

Another approach is the one presented in [104]. Its main idea is to approximate the union of local coresets with another coreset. They assume nodes communicate over a rooted tree, with each node passing its coreset to its parent. Because the approximation factor of the constructed coreset depends on the quality of its component coresets, the accuracy a coreset needs (and thus the overall communication complexity) scales with the height of this tree. Although it is possible to find a spanning tree in any communication network, when the graph has large diameter every tree has large height. In particular many natural networks such as grid networks have a large diameter ($\Omega(\sqrt{s})$ for grids) which greatly increases the size of coresets which must be communicated across the lower levels of the tree. We show that it is possible to construct a global coreset with low communication overhead. This is done by distributing the coreset construction procedure rather than combining local coresets. The communication needed to construct this coreset is negligible – just a single value from each data set representing the approximate cost of their local optimal clustering. Since the sampled global ϵ-coreset is the same size as any local ϵ-coreset, this leads to an improvement of the communication cost over the other approaches. See Figure 10 for an illustration. The constructed coreset is smaller by a factor of n in general graphs, and is independent of the communication topology. This method excels in sparse networks with large diameters, where the previous approach in [104] requires



(a) Zhang et al.[104]

(b) Our Construction

Figure 10: Illustration of different coreset construction approaches.

coresets that are quadratic in the size of the diameter for k-median and quartic for k-means; see Section 3.2.2 for details. [47] also merge coresets using coreset construction, but they do so in a model of parallel computation and ignore communication costs.

Additional related work. Many empirical algorithms adapt the centralized algorithms to the distributed setting. They generally provide no bound for the clustering quality or the communication cost. For instance, a technique is proposed in [48] to adapt several iterative center-based data clustering algorithms including Lloyd's algorithm for k-means to the distributed setting, where sufficient statistics instead of the raw data are sent to a central coordinator. This approach involves transferring data back and forth in each iteration, and thus the communication cost depends on the number of iterations. Similarly, the communication costs of the distributed clustering algorithms proposed in [38] and [101] depend on the number of iterations. Some other algorithms gather local summaries and then perform global clustering on the summaries. The distributed density-based clustering algorithm in [60] clusters and computes summaries for the local data at each node, and sends the local summaries

to a central node where the global clustering is carried out. This algorithm only considers the flat two-tier topology. Some in-network aggregation schemes for computing statistics over distributed data are useful for such distributed clustering algorithms. For example, an algorithm is provided in [35] for approximate duplicate-sensitive aggregates across distributed data sets, such as SUM. An algorithm is proposed in [54] for power-preserving computation of order statistics such as quantile.

Several coreset construction algorithms have been proposed for k-median, k-means and k-line median clustering [57, 30, 56, 74, 45]. For example, the algorithm in [45] constructs a coreset of size $\tilde{O}(kd/\epsilon^2)$ whose cost approximates that of the original data up to accuracy ϵ with respect to k-median in $\mathbf{R}^d$. All of these algorithms consider coreset construction in the centralized setting, while our construction algorithm is for the distributed setting.

There has also been work attempting to parallelize clustering algorithms. [47] showed that coresets could be constructed in parallel and then merged together. Bahmani et al. [12] adapted `k-means++` to the parallel setting. Their algorithm, k-means||, essentially builds $O(1)$-coreset of size $O(k \log |P|)$. However, it cannot build ϵ-coreset for $\epsilon = o(1)$, and thus can only guarantee constant approximation solutions.

There is also related work providing approximation solutions for k-median based on random sampling [25]. Particularly, they showed that given a sample of size $\tilde{O}(\frac{k}{\epsilon^2})$ drawn i.i.d. from the data, there exists an algorithm that outputs a solution with an average cost bounded by twice the optimal average cost plus an error bound ϵ. If we convert it to a multiplicative approximation factor, the factor depends on the optimal average cost. When there are outlier points far away from all other points, the optimal average cost can be very small after normalization, then the multiplicative approximation factor is large. The coreset approach provides better guarantees. Additionally, their approach is not applicable to k-means.

Balcan et al. [18] and Daume et al. [39] consider fundamental communication

complexity questions arising when doing classification in distributed settings. In concurrent and independent work, Vempala et al. [62] study several optimization problems in distributed settings, including k-means clustering under an interesting separability assumption.

3.2.1 Distributed Coreset Construction

Here we design a distributed coreset construction algorithm for k-means and k-median. Note that the underlying technique can be extended to other additive clustering objectives such as k-line median and sum of distances between points and their centers to the power of z.

To gain some intuition on the distributed coreset construction algorithm, we briefly review the coreset construction algorithm in [45] in the centralized setting. The coreset is constructed by computing a constant approximation solution for the entire data set, and then sampling points proportional to their contributions to the cost of this solution. Intuitively, the points close to the nearest centers can be approximately represented by the nearest centers while points far away cannot be well represented. Thus, points should be sampled with probability proportional to their contributions to the cost.

Directly adapting the algorithm to the distributed setting would require computing a constant approximation solution for the entire data set. We show that a global coreset can be constructed in a distributed fashion by estimating the weight of the entire data set with the sum of local approximations. We first compute a local approximation solution for each local data set, and communicate the total costs of these local solutions. Then we sample points proportional to their contributions to the cost of their local solutions. At the end of the algorithm, the coreset consists of the sampled points and the centers in the local solutions. The coreset points are distributed over the nodes, so we call it distributed coreset. See Algorithm 8 for details.

Algorithm 8 Communication aware distributed coreset construction

Input: Local data sets $\{P_i\}_{i=1}^s$, parameter t (number of points to be sampled).

1: **for** each node $v_i \in V$ **do**
2: Compute a constant approximation B_i for P_i.
3: Communicate $\text{cost}(P_i, B_i)$ to all other nodes.
4: **end for**
5: **for** each node $v_i \in V$ **do**
6: Set $t_i = \frac{t \ \text{cost}(P_i, B_i)}{\sum_{j=1}^s \text{cost}(P_j, B_j)}$. Set $m_p = \text{cost}(p, B_i)$ for each $p \in P_i$.
7: Pick a non-uniform random sample S_i of t_i points from P_i, where for every $q \in S_i$ and $p \in P_i$, we have $q = p$ with probability $m_p / \sum_{z \in P_i} m_z$.
8: Let $w_q = \sum_i \sum_{z \in P_i} m_z / (t m_q)$ for each $q \in S_i$.
9: **for** each $b \in B_i$ **do**
10: Let $P_b = \{p \in P_i : d(p, b) = d(p, B_i)\}$, $w_b = |P_b| - \sum_{q \in P_b \cap S} w_q$.
11: **end for**
12: **end for**

Output: Distributed coreset: points $S_i \cup B_i$ with weights $\{w_q : q \in S_i \cup B_i\}_{i=1}^s$.

Theorem 10. *For distributed k-means and k-median clustering on a graph, there exists an algorithm such that with probability at least $1 - \delta$, the union of its output on all nodes is an ϵ-coreset for $P = \cup_{i=1}^s P_i$. The size of the coreset is $O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta})$ for k-means, and $O(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta}) + nk)$ for k-median. The total communication cost is $O(mn)$.*

As described below, the distributed coreset construction can be achieved by using Algorithm 8 with appropriate t, namely $O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta})$ for k-means and $O(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta}))$ for k-median. The formal proofs are described in the following subsections.

3.2.1.1 Proof of Theorem 14: k-median

The analysis relies on the definition of the pseudo-dimension of a function space and a sampling lemma.

Definition 11 ([77, 45])**.** *Let F be a finite set of functions from a set P to $\mathbf{R}_{\geq 0}$. For $f \in F$, let $B(f, r) = \{p : f(p) \leq r\}$. The dimension of the function space* $\dim(F, P)$ *is the smallest integer d such that for any $G \subseteq P$, $\left|\{G \cap B(f, r) : f \in F, r \geq 0\}\right| \leq |G|^d$.*

Suppose we draw a sample S according to $\{m_p : p \in P\}$, namely for every $q \in S$ and every $p \in P$, we have $q = p$ with probability $\frac{m_p}{\sum_{z\in P} m_z}$. Set the weights of the points as $w_p = \frac{\sum_{z\in P} m_z}{m_p|S|}$ for $p \in P$. Then for any $f \in F$, the expectation of the weighted cost of S equals the cost of the original data P:

$$\begin{aligned}\mathbf{E}\left[\sum_{q\in S} w_q f(q)\right] &= \sum_{q\in S}\mathbf{E}[w_q f(q)] = \sum_{q\in S}\sum_{p\in P}\Pr[q=p]w_p f(p)\\ &= \sum_{q\in S}\sum_{p\in P}\frac{m_p}{\sum_{z\in P} m_z}\frac{\sum_{z\in P} m_z}{m_p|S|}f(p) = \sum_{q\in S}\sum_{p\in P}\frac{1}{|S|}f(p) = \sum_{p\in P} f(p).\end{aligned}$$

The following lemma shows that if the sample size is large enough, then we also have concentration for any $f \in F$. The lemma is implicit in [45] and we include the proof in the appendix for completeness.

Lemma 16. *Fix a set F of functions $f : P \to \mathbf{R}_{\geq 0}$. Let S be a sample drawn i.i.d. from P according to $\{m_p : p \in P\}$, namely, for every $q \in S$ and every $p \in P$, we have $q = p$ with probability $\frac{m_p}{\sum_{z\in P} m_z}$. Let $w_p = \frac{\sum_{z\in P} m_z}{m_p|S|}$ for $p \in P$. For a sufficiently large c, if $|S| \geq \frac{c}{\epsilon^2}\left(\dim(F,P) + \log\frac{1}{\delta}\right)$ then with probability at least $1-\delta, \forall f \in F$: $\left|\sum_{p\in P} f(p) - \sum_{q\in S} w_q f(q)\right| \leq \epsilon\left(\sum_{p\in P} m_p\right)\left(\max_{p\in P}\frac{f(p)}{m_p}\right)$.*

To get a small bound on the difference between $\sum_{p\in P} f(p)$ and $\sum_{q\in S} w_q f(q)$, we need to choose m_p such that $\max_{p\in P}\frac{f(p)}{m_p}$ is bounded. More precisely, if we choose $m_p = \max_{f\in F} f(p)$, then the difference is bounded by $\epsilon\sum_{p\in P} m_p$.

We first consider the centralized setting and review how [45] applied the lemma to construct a coreset for k-median as in Definition 10. A natural approach is to apply this lemma directly to the cost, namely, to choose $f_{\mathbf{x}}(p) := \text{cost}(p, \mathbf{x})$. The problem is that a suitable upper bound m_p is not available for $\text{cost}(p, \mathbf{x})$. However, we can still apply the lemma to a different set of functions defined as follows. Let b_p denote the closest center to p in the approximation solution. Aiming to approximate the error $\sum_p[\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})]$ rather than to approximate $\sum_p \text{cost}(p, \mathbf{x})$ directly, we define $f_{\mathbf{x}}(p) := \text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + \text{cost}(p, b_p)$, where $\text{cost}(p, b_p)$ is added so that

$f_{\mathbf{x}}(p) \geq 0$. Since $0 \leq f_{\mathbf{x}}(p) \leq 2\text{cost}(p, b_p)$, we can apply the lemma to $f_{\mathbf{x}}(p)$ and $m_p = 2\text{cost}(p, b_p)$. The lemma then bounds the difference $|\sum_{p\in P} f_{\mathbf{x}}(p) - \sum_{q\in S} w_q f_{\mathbf{x}}(q)|$ by $2\epsilon \sum_{p\in P} \text{cost}(p, b_p)$, so we have an $O(\epsilon)$-approximation.

Note that $\sum_{p\in P} f_{\mathbf{x}}(p) - \sum_{q\in S} w_q f_{\mathbf{x}}(q)$ does not equal $\sum_{p\in P} \text{cost}(p, \mathbf{x}) - \sum_{q\in S} w_q \text{cost}(q, \mathbf{x})$. However, it equals the difference between $\sum_{p\in P} \text{cost}(p, \mathbf{x})$ and a weighted cost of the sampled points and the centers in the approximation solution. To get a coreset as in Definition 10, we need to add the centers of the approximation solution with specific weights to the coreset. Then when the sample is sufficiently large, the union of the sampled points and the centers is an ϵ-coreset.

Our key contribution in this paper is to show that in the distributed setting, it suffices to choose b_p from the local approximation solution for the local data set containing p, rather than from an approximation solution for the global data set. Furthermore, the sampling and the weighting of the coreset points can be done in a local manner. In the following, we provide a formal verification of our discussion above. We have the following lemma for k-median with $F = \{f_{\mathbf{x}} : f_{\mathbf{x}}(p) = d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + d(p, b_p), \mathbf{x} \in (\mathbf{R}^d)^k\}$.

Lemma 17. *For k-median, the output of Algorithm 8 is an ϵ-coreset with probability at least $1 - \delta$, if $t \geq \frac{c}{\epsilon^2}\left(\dim(F, P) + \log\frac{1}{\delta}\right)$ for a sufficiently large constant c.*

Proof. We want to show that for any set of centers $\mathbf{x}$ the true cost for using these centers is well approximated by the cost on the weighted coreset. Note that our coreset has two types of points: sampled points $p \in S = \cup_{i=1}^{s} S_i$ with weight $w_p := \frac{\sum_{z\in P} m_z}{m_p |S|}$ and local solution centers $b \in B = \cup_{i=1}^{s} B_i$ with weight $w_b := |P_b| - \sum_{p\in S\cap P_b} w_p$. We use b_p to represent the nearest center to p in the local approximation solution. We use P_b to represent the set of points having b as their closest center in the local approximation solution.

As mentioned above, we construct $f_{\mathbf{x}}$ to be the difference between the cost of p and the cost of b_p on $\mathbf{x}$ so that Lemma 16 can be applied to $f_{\mathbf{x}}$. Note that $0 \leq f_{\mathbf{x}}(p) \leq$

$2d(p, b_p)$ by triangle inequality, and S is sufficiently large and chosen according to weights $m_p = d(p, b_p)$, so the conditions of Lemma 16 are met. Then we have

$$\begin{aligned} D =& \left| \sum_{p \in P} f_{\mathbf{x}}(p) - \sum_{q \in S} w_q f_{\mathbf{x}}(q) \right| \leq 2\epsilon \sum_{p \in P} m_p = 2\epsilon \sum_{p \in P} d(p, b_p) = 2\epsilon \sum_{i=1}^{s} d(P_i, B_i) \\ \leq & O(\epsilon) \sum_{p \in P} d(p, \mathbf{x}) \end{aligned}$$

where the last inequality follows from the fact that B_i is a constant approximation solution for P_i.

Next, we show that the coreset is constructed such that D is exactly the difference between the true cost and the weighted cost of the coreset, which then leads to the lemma.

Note that the centers are weighted such that

$$\sum_{b \in B} w_b d(b, \mathbf{x}) = \sum_{b \in B} |P_b| d(b, \mathbf{x}) - \sum_{b \in B} \sum_{q \in S \cap P_b} w_q d(b, \mathbf{x}) = \sum_{p \in P} d(b_p, \mathbf{x}) - \sum_{q \in S} w_q d(b_q, \mathbf{x}). \tag{13}$$

Also note that $\sum_{p \in P} m_p = \sum_{q \in S} w_q m_q$, so

$$\begin{aligned} D =& \left| \sum_{p \in P} \left[d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + m_p \right] - \sum_{q \in S} w_q \left[d(q, \mathbf{x}) - d(b_q, \mathbf{x}) + m_q \right] \right| \\ =& \left| \sum_{p \in P} d(p, \mathbf{x}) - \sum_{q \in S} w_q d(q, \mathbf{x}) - \left[\sum_{p \in P} d(b_p, \mathbf{x}) - \sum_{q \in S} w_q d(b_q, \mathbf{x}) \right] \right|. \end{aligned} \tag{14}$$

By plugging (13) into (14), we have

$$D = \left| \sum_{p \in P} d(p, \mathbf{x}) - \sum_{q \in S} w_q d(q, \mathbf{x}) - \sum_{b \in B} w_b d(b, \mathbf{x}) \right| = \left| \sum_{p \in P} d(p, \mathbf{x}) - \sum_{q \in S \cup B} w_q d(q, \mathbf{x}) \right|$$

which implies the lemma. □

In [45] it is shown that [3] $\dim(F, P) = O(kd)$. Therefore, by Lemma 17, when $|S| \geq O\left(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta})\right)$, the weighted cost of $S \cup B$ approximates the k-median

[3] For both k-median and k-means in general metric spaces, $\dim(F, P) = O(k \log |P|)$, so the bound for general metric spaces (including Euclidean space we focus on) can be obtained by replacing d with $\log |P|$.

cost of P for any set of centers, then $(S \cup B, w)$ is an ϵ-coreset for P. The total communication cost is bounded by $O(mn)$, since even in the most general case when every node only knows its neighbors, we can broadcast the local costs with $O(mn)$ communication (see Algorithm 10).

3.2.1.2 Proof of Theorem 14: k-means

We have for k-means a similar lemma that when $t = O(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$, the algorithm constructs an ϵ-coreset with probability at least $1-\delta$. The key idea is the same as that for k-median: we use centers b_p from the local approximation solutions as an approximation to the original data points p, and show that the error between the total cost and the weighted sample cost is approximately the error between the cost of p and its sampled cost (compensated by the weighted centers), which is shown to be small by Lemma 16.

The key difference between k-means and k-median is that triangle inequality applies directly to the k-median cost. In particular, for the k-median problem note that $\text{cost}(b_p, p) = d(b_p, p)$ is an upper bound for the error of b_p on any set of centers, i.e. $\forall \mathbf{x} \in (\mathbf{R}^d)^k$, $d(b_p, p) \geq |d(p, \mathbf{x}) - d(b_p, \mathbf{x})| = |\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})|$ by triangle inequality. Then we can construct $f_{\mathbf{x}}(p) := \text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + d(b_p, p)$ such that $h_p(\mathbf{x})$ is bounded. In contrast, for k-means, the error $|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| = |d(p, \mathbf{x})^2 - d(b_p, \mathbf{x})^2|$ does not have such an upper bound. The main change to the analysis is that we divide the points into two categories: good points whose costs approximately satisfy the triangle inequality (up to a factor of $1/\epsilon$) and bad points. The good points for a fixed set of centers $\mathbf{x}$ are defined as

$$G(\mathbf{x}) = \{p \in P : |\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| \leq \Delta_p\}$$

where the upper bound is $\Delta_p = \frac{\text{cost}(p, b_p)}{\epsilon}$. Good points we can bound as before. For bad points we can show that while the difference in cost may be larger than $\text{cost}(p, b_p)/\epsilon$, it must still be small, namely $O(\epsilon \min\{\text{cost}(p, \mathbf{x}), \text{cost}(b_p, \mathbf{x})\})$.

Formally, the functions $f_{\mathbf{x}}(p)$ are restricted to be defined only over good points:

$$f_{\mathbf{x}}(p) = \begin{cases} \text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + \Delta_p & \text{if } p \in G(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

Then $\sum_{p \in P} \text{cost}(p, \mathbf{x}) - \sum_{q \in S \cup B} w_q \text{cost}(q, \mathbf{x})$ is decomposed into three terms:

$$\sum_{p \in P} f_{\mathbf{x}}(p) - \sum_{q \in S} w_q f_{\mathbf{x}}(q) \tag{15}$$

$$+ \sum_{p \in P \setminus G(\mathbf{x})} [\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + \Delta_p] \tag{16}$$

$$- \sum_{q \in S \setminus G(\mathbf{x})} w_q [\text{cost}(q, \mathbf{x}) - \text{cost}(b_q, \mathbf{x}) + \Delta_q] \tag{17}$$

Lemma 16 bounds (15) by $O(\epsilon)\text{cost}(P, \mathbf{x})$, but we need an accuracy of ϵ^2 to compensate for the $1/\epsilon$ factor in the upper bound, resulting in a $O(1/\epsilon^4)$ factor in the sample complexity.

We begin by bounding (16). Note that for each term in (16), $|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| > \Delta_p$ since $p \notin G(\mathbf{x})$. Furthermore, $p \notin G(\mathbf{x})$ only when p and b_p are close to each other and far away from $\mathbf{x}$. In Lemma 25 we use this to show that $|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| \leq O(\epsilon) \min\{\text{cost}(p, \mathbf{x}), \text{cost}(b_p, \mathbf{x})\}$. The details are presented in the appendix.

Using Lemma 25, (16) can be bounded by $O(\epsilon) \sum_{p \in P \setminus G(\mathbf{x})} \text{cost}(p, \mathbf{x}) \leq O(\epsilon)\text{cost}(P, \mathbf{x})$.

Similarly, by the definition of Δ_q and Lemma 25, (17) is bounded by

$$\begin{aligned} (17) &\leq \sum_{q \in S \setminus G(\mathbf{x})} 2w_q |\text{cost}(q, \mathbf{x}) - \text{cost}(b_q, \mathbf{x})| \leq O(\epsilon) \sum_{q \in S \setminus G(\mathbf{x})} w_q \text{ cost}(b_q, \mathbf{x}) \\ &\leq O(\epsilon) \sum_{b \in B} \left(\sum_{q \in P_b \cap S} w_q \right) \text{cost}(b, \mathbf{x}). \end{aligned}$$

Note that the expectation of $\sum_{q \in P_b \cap S} w_q$ is $|P_b|$. By a sampling argument (Lemma 26), if $t \geq O(nk \log \frac{nk}{\delta})$, then $\sum_{q \in P_b \cap S} w_q \leq 2|P_b|$. Then (17) is bounded by

$$O(\epsilon) \sum_{b \in B} \text{cost}(b, \mathbf{x}) |P_b| = O(\epsilon) \sum_{p \in P} \text{cost}(b_p, \mathbf{x})$$

where $\sum_{p \in P} \text{cost}(b_p, \mathbf{x})$ is at most a constant factor more than the optimum cost.

Since each of (15),(16), and (17) is $O(\epsilon)\text{cost}(P, \mathbf{x})$, we know that their sum is the same magnitude. Combining the above bounds, we have

$$\left| \text{cost}(P, \mathbf{x}) - \sum_{q \in S \cup B} w_q \text{cost}(q, \mathbf{x}) \right| \leq O(\epsilon)\text{cost}(P, \mathbf{x}).$$

The proof is then completed by choosing a suitable ϵ, and bounding $\dim(F, P) = O(kd)$ as in [45].

3.2.2 Effect of Network Topology on Communication Cost

In the previous section, we presented a distributed coreset construction algorithm. The coreset constructed can then be used as a proxy for the original data, and we can run any distributed clustering algorithm on it. In this paper, we discuss the approach of simply collecting all local portions of the distributed coreset and run non-distributed clustering algorithm on it. If there is a central coordinator in the communication graph, then we can simply send the local portions of the coreset to the coordinator which can perform the clustering task. The total communication cost is just the size of the coreset.

Here we consider the distributed clustering tasks where the nodes are arranged in some arbitrary connected topology, and can only communicate with their neighbors. We propose a message passing approach for globally sharing information, and use it for collecting information for coreset construction and sharing the local portions of the coreset. We also consider the special case when the graph is a rooted tree.

3.2.2.1 General Graphs

We now present the main result for distributed clustering on graphs.

Theorem 11. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least*

Algorithm 9 Distributed clustering on a graph

Input: $\{P_i, 1 \le i \le s\}$: local data sets; $\{N_i, 1 \le i \le s\}$: the neighbors of v_i; $\mathcal{A}_\alpha$: an α-approximation algorithm for weighted clustering instances.

1: **for** each node $v_i \in V$ **do**
2: Construct its local portion D_i of an $\epsilon/2$-coreset by Algorithm 8, using Message-Passing for communicating the local costs.
3: **end for**
4: **for** each node $v_i \in V$ **do**
5: Call Message-Passing(D_i, N_i). Let $\mathbf{x} = \mathcal{A}_\alpha(\cup_j D_j)$.
6: **end for**

Output: $\mathbf{x}$.

Algorithm 10 Message-Passing(I_i, N_i)

Input: I_i is the message, N_i are the neighbors.

1: Let R_i denote the information received.
2: Initialize $R_i = \{I_i\}$, and send I_i to all the neighbors.
3: **while** $R_i \neq \{I_j, 1 \le j \le s\}$ **do**
4: **if** receive message $I_j \notin R_i$ **then**
5: $R_i = R_i \cup \{I_j\}$ and send I_j to all the neighbors.
6: **end if**
7: **end while**

$1-\delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively) clustering. The total communication cost is $O(m(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$ for k-means, and $O(m(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ for k-median.

Proof. The details are presented in Algorithm 9. By Theorem 14, the output of Algorithm 8 is a coreset. Observe that in Algorithm 10, for any j, I_j propagates on the graph in a breadth-first-search style, so at the end every node receives I_j. This holds for all $1 \le j \le s$, so all nodes has a copy of the coreset at the end, and thus the output is a $(1+\epsilon)\alpha$-approximation solution.

Also observe that in Algorithm 10, for any node v_i and $j \in [s]$, v_i sends out I_j once, so the communication of v_i is $|N_i| \times \sum_{j=1}^{s} |I_j|$. The communication cost of Algorithm 10 is $O(m\sum_{j=1}^{s}|I_j|)$. Then the total communication cost of Algorithm 9 follows from the size of the coreset constructed. □

In contrast, an approach where each node constructs an ϵ-coreset for k-means and

sends it to the other nodes incurs communication cost of $\tilde{O}(\frac{mnkd}{\epsilon^4})$. Our algorithm significantly reduces this.

3.2.2.2 Rooted Trees

Our algorithm can also be applied on a rooted tree, and compares favorably to other approaches involving coresets [104]. We can restrict message passing to operating along this tree, leading to the following theorem.

Theorem 12. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least $1-\delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively) clustering on a rooted tree of height h. The total communication cost is $O(h(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$ for k-means, and $O(h(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ for k-median.*

Proof. We can construct the distributed coreset using Algorithm 8. In the construction, the costs of the local approximation solutions are sent from every node to the root, and the sum is sent to every node by the root. After the construction, the local portions of the coreset are sent from every node to the root. A local portion D_i leads to a communication cost of $O(|D_i|h)$, so the total communication cost is $O(h\sum_{i=1}^{s}|D_i|)$. Once the coreset is constructed at the root, the α-approximation algorithm can be applied centrally, and the results can be sent back to all nodes. □

Our approach improves the cost of $\tilde{O}(\frac{nh^4kd}{\epsilon^4})$ for k-means and the cost of $\tilde{O}(\frac{nh^2kd}{\epsilon^2})$ for k-median in [104] [4]. The algorithm in [104] builds on each node a coreset for the union of coresets from its children, and thus needs $O(\epsilon/h)$ accuracy to prevent the

[4] Their algorithm used coreset construction as a subroutine. The construction algorithm they used builds coreset of size $\tilde{O}(\frac{nkh}{\epsilon^d}\log|P|)$. Throughout this paper, when we compare to [104] we assume they use the coreset construction technique of [45] to reduce their coreset size and communication cost.

accumulation of errors. Since the coreset construction subroutine has quadratic dependence on $1/\epsilon$ for k-median (quartic for k-means), the algorithm then has quadratic dependence on h (quartic for k-means). Our algorithm does not build coreset on top of coresets, resulting in a better dependence on the height of the tree h.

In a general graph, any rooted tree will have its height h at least as large as half the diameter. For sensors in a grid network, this implies $h = \Omega(\sqrt{s})$. In this case, our algorithm gains a significant improvement over existing algorithms.

3.3 Distributed k-Means Clustering of High Dimensional Data

In Algorithm 8, the number of points in the coreset is independent of the number of the original data points, which is useful for large-scale applications. However, it is linear in the dimension of the data, leading to high communication cost for high dimensional data. Here we propose a distributed PCA algorithm, and show that its output represents the original data in the sense that any good approximation solution of k-means clustering on the output projected data is also a good solution on the original data. When combined with the distributed coreset approach, this leads to an algorithm whose communication cost (in terms of the number of points communicated) is independent of the size and the dimension of the original data.

Related Work on Distributed PCA. Algorithms for (approximate) distributed PCA have been proposed [93, 13, 75, 83, 46], but either without theoretical guarantees on the solution quality, or without considerations of the communication cost. Most closely related to our work is [46], which pointed out that the top singular vectors and eigenvalues of the local data set can be viewed as its summary and the union of the local summaries can be viewed as a summary of the global data. It implicitly showed the correctness of the method, but without considering the communication. [93] proposed a variant of the algorithm but provided no theoretical analysis on the tradeoff between communication and approximation guarantees.

In [62] the authors study efficient algorithms in the distributed model. Their model is different, namely, it is an arbitrary partition model in which each server holds a matrix P_i and $P = \sum_{i=1}^{s} P_i$. Thus, each row of P is additively shared across the s servers, whereas in our model each row of P belongs to a single server, though duplicate rows are allowed. Our model is motivated by applications in which points are indecomposable entities, such as the clustering application. We also focus on how the distributed PCA algorithm affects the quality of the final clustering solution, while [62] focus on getting a low rank approximation.

Other related work includes the recent [50] (see also the references therein), who give a deterministic streaming algorithm for approximate PCA in which each point of P is seen one at a time and uses $O(dk/\epsilon)$ words of communication. Their algorithm naturally gives an $O(sdk/\epsilon)$ communication algorithm in the distributed model. However, it involves an SVD computation for each point, making the overall computation expensive.

3.3.1 Distributed PCA

Our distributed PCA algorithm is described in Algorithm 11, where BROADCAST is a shorthand for communicating information to all other nodes. The algorithm performs local PCA on each local data set, and communicates the t largest principal components. These are then concatenated and used to get the t largest global principal components. Finally, all the local data are projected on these t global principal components. See Figure 11 for an illustration.

$$P = \begin{bmatrix} P_1 \\ \vdots \\ P_s \end{bmatrix} \begin{matrix} \xrightarrow{\text{Local PCA}} \\ \vdots \\ \xrightarrow{\text{Local PCA}} \end{matrix} \begin{bmatrix} D_1^{(t_1)} \left(V_1^{(t)}\right)^\top \\ \vdots \\ D_s^{(t_1)} \left(V_s^{(t)}\right)^\top \end{bmatrix} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_s \end{bmatrix} = Y \xrightarrow{\text{Global PCA}} V^{(t)}$$

Figure 11: The key points of the distributed PCA algorithm.

Algorithm 11 Distributed PCA

Input: local data $\{P_i\}_{i=1}^s$ and parameter $t \in \mathbf{N}_+$.

1: **for** each node $v_i \in V$ **do**
2: Compute local SVD: $P_i = U_i D_i V_i^\top$.
3: BROADCAST $D_i^{(t)}, V_i^{(t)}$.
4: **end for**
5: **for** each node $v_i \in V$ **do**
6: Set $Y_i = D_i^{(t)}(V_i^{(t)})^\top$, $Y^\top = [Y_1^\top, \ldots, Y_s^\top]$.
7: Compute global SVD: $Y = UDV^\top$.
8: Compute projected data: $\hat{P}_i = P_i^{(t)} V^{(t)} (V^{(t)})^\top$.
9: **end for**

Output: $\{\hat{P}_i\}_{i=1}^s$.

Here we provide a theoretical analysis, which leads to a way to set the algorithm parameters, so that we will not compromise much on the quality of the clustering obtained on the projected data. In the following, we always assume for simplicity that P is normalized, that is, $\sum_{p \in P} p = 0$. We now show that the output of Algorithm 11 approximates the original data in the sense that their distances to low dimension subspaces are almost the same:

Theorem 13. *Let X be a $d \times j$ matrix whose columns are orthonormal. Let $\epsilon \in (0, 1]$ and $t \in \mathbf{N}$ with $d - 1 \geq t \geq j + \lceil 8j/\epsilon \rceil - 1$. Then the output of Algorithm 11 satisfies*

$$0 \leq \|PX\|_F^2 - \|\hat{P}X\|_F^2 \leq \epsilon d^2(P, L(X)) \quad \text{and} \quad 0 \leq \|PX - \hat{P}X\|_F^2 \leq \epsilon d^2(P, L(X)).$$

Intuitively, it implies that the squared distances to any low dimension subspace $L(X)$ from the projected data and the original data are approximately equal when the number of principal components used is sufficiently large compared to the dimension of $L(X)$. As shown in the next section, this guarantees that the projected data can act as a proxy for the original data in k-means clustering.

The key in the proof of the theorem is a property about SVD: the projection $\tilde{A}$ of a $n \times d$ matrix A to the subspace spanned by its first t right singular vectors approximates A in the following sense.

Lemma 18. *Let $A \in \mathbf{R}^{n\times d}$ be an $n \times d$ matrix with singular value decomposition $A = UDV^\top$. Let $\epsilon \in (0, 1]$ and $r, t \in \mathbf{N}_+$ with $d - 1 \geq t \geq r + \lceil r/\epsilon \rceil - 1$, and let $\tilde{A} = AV^{(t)}(V^{(t)})^\top$. Then for any matrix X with d rows and $\|X\|_F^2 \leq r$, we have*

$$\|(A - \tilde{A})X\|_F^2 = \|AX\|_F^2 - \|\tilde{A}X\|_F^2 \leq \epsilon \sum_{i=r+1}^{d} \sigma_i^2(A).$$

A special important case is that X is the orthonormal basis for r-dimensional subspace, when the lemma reduces to Lemma 6.1 in [46]. In this case, the lemma means that the projections of $\tilde{A}$ and A on any r-dimensional subspace are close, when the projected dimension t is sufficiently large compared to r. The proof of the theorem will need a statement in the more general setting where X may not be orthonormal. *Proof Sketch of Theorem 13.* We first introduce some auxiliary variables for the analysis, which act as intermediate connections between P and $\hat{P}$. Imagine we perform two kinds of projections: first project P_i to $\tilde{P}_i = P_i V_i^{(t)}(V_i^{(t)})^\top$, then project $\tilde{P}_i$ to $\overline{P}_i = \tilde{P}_i V^{(t)}(V^{(t)})^\top$. Let $\tilde{P}$ denote the vertical concatenation of $\tilde{P}_i$ and let $\overline{P}$ denote the vertical concatenation of $\overline{P}_i$, i.e.

$$\tilde{P} = \begin{bmatrix} \tilde{P}_1 \\ \vdots \\ \tilde{P}_s \end{bmatrix} \quad \text{and} \quad \overline{P} = \begin{bmatrix} \overline{P}_1 \\ \vdots \\ \overline{P}_s \end{bmatrix}$$

These variables are designed such that the difference between P and $\overline{P}$ is easily bounded. Our proof proceeds by first bounding the diference between P and $\overline{P}$, and then bounding that between $\overline{P}$ and $\hat{P}$.

Consider $\|PX\|_F^2 - \|\hat{P}X\|_F^2$. It can be decomposed as followings:

$$\begin{aligned} \|PX\|_F^2 - \|\hat{P}X\|_F^2 &= \left[\|PX\|_F^2 - \|\tilde{P}X\|_F^2\right] + \left[\|\tilde{P}X\|_F^2 - \|\overline{P}X\|_F^2\right] \\ &+ \left[\|\overline{P}X\|_F^2 - \|\hat{P}X\|_F^2\right]. \end{aligned}$$

The first term $\|PX\|_F^2 - \|\tilde{P}X\|_F^2 = \sum_{i=1}^{s}\left[\|P_iX\|_F^2 - \|\tilde{P}_iX\|_F^2\right]$, each of which can be bounded by Lemma 18 since $\tilde{P}_i$ is the SVD truncation of P. The second term

can be bounded similarly. The more difficult part is to bound the third term. Let $Z = V^{(t)}(V^{(t)})^\top X$. Then by definition, $\overline{P}_i = \tilde{P}_i Z$, $\hat{P}_i = P_i Z$, and

$$\|\overline{P}X\|_F^2 - \|\hat{P}X\|_F^2 = \sum_{i=1}^{s} \left[\|\tilde{P}_i Z\|_F^2 - \|P_i Z\|_F^2 \right].$$

Then Lemma 18 can be applied to show that $\|\tilde{P}_i Z\|_F^2 - \|P_i Z\|_F^2$ is indeed small.

The bound on $\|PX - \hat{P}X\|_F^2$ can also be proved by a similar argument. The complete proof can be found in Appendix B.3. □

3.3.2 Distributed Clustering

In this subsection, we show that any good approximation solution on the projected data constructed by the distributed PCA algorithm is also a good approximation on the original data.

Theorem 14. *Let* $\mathbf{x}$ *be a set of* k *centers in* $\mathbf{R}^d$. *Let* $\epsilon \in (0, 1]$ *and* $t \in \mathbf{N}$ *with* $d - 1 \geq t \geq k + \lceil 50k/\epsilon^2 \rceil$. *Then there exists a constant* $c_0 \geq 0$, *such that the output of Algorithm 11 satisfies* $(1-\epsilon)\text{cost}(P, \mathbf{x}) \leq \text{cost}(\hat{P}, \mathbf{x}) + c_0 \leq (1+\epsilon)\text{cost}(P, \mathbf{x})$.

The analysis follows the ideas in [46]. Let $X \in \mathbf{R}^{d \times k}$ has orthonormal columns that span $\mathbf{x}$. The cost of P can be decomposed into two parts: the squared distances $d^2(P, L(X))$ to the subspace spanned by X, and the squared distances $\sum_i d^2(p_X(p_i), \mathbf{x})$ between the projection of the points on $L(X)$ and $\mathbf{x}$. The cost of $\hat{P}$ can be decomposed similarly. Their difference in the first part (compensated by $c_0 = \|P\|_F^2 - \|\hat{P}\|_F^2$) can be bounded by $\|PX\|_F^2 - \|\hat{P}X\|_F^2$. The difference in the second part can be bounded approximately by $\sum_i d^2(p_X(p_i), p_X(\hat{p}_i))/\epsilon = \|PX - \hat{P}X\|_F^2/\epsilon$. Then the theorem follows from Theorem 13. The complete proof is provided in the appendix.

By Theorem 14, the distributed coreset construction algorithm in the last section can be applied on the projected data to get a coreset of size independent of the original dimension. Then we get an algorithm with low communication cost for high dimensional data, which is summarized in Theorem 15.

Theorem 15. *Given an α-approximation algorithm for k-means as a subroutine, there exists an algorithm that with probability at least $1-\delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means clustering. The total communication cost is $O(\frac{msk}{\epsilon^2})$ vectors in $\mathbf{R}^d$ plus $O\left(\frac{m}{\epsilon^4}(\frac{k^2}{\epsilon^2}+\log\frac{1}{\delta})+msk\log\frac{sk}{\delta}\right)$ vectors in $\mathbf{R}^{O(k/\epsilon^2)}$.*

3.4 Experiments

The empirical study consists of two sets of experiments. The first set evaluates the coreset-based algorithm in Section 3.2, and the second set evaluates the approach in Section 3.3 that first reduces data dimension by distributed PCA and then applies the coreset-based algorithm.

3.4.1 Experiments on Distributed Coreset-Based Clustering

In these experiments, we seek to determine whether our algorithm is effective for the clustering tasks and how it compares to the other distributed coreset algorithms [5]. We present the k-means cost of the solution produced by our algorithm with varying communication cost, and compare to those of other algorithms when they use the same amount of communication.

Data sets. Following the setup of [104, 12], for the synthetic data we randomly choose $k=5$ centers from the standard Gaussian distribution in $\mathbf{R}^{10}$, and sample equal number of $20,000$ points from the Gaussian distribution around each center. Note that, as in [104, 12], we use the cost of the centers as a baseline for comparing the clustering quality. We choose the following real world data sets from [11]: Spam (4601 points in $\mathbf{R}^{58}$), Pendigits (10992 points in $\mathbf{R}^{16}$), Letter (20000 points in $\mathbf{R}^{16}$), and ColorHistogram of the Corel Image data set (68040 points in $\mathbf{R}^{32}$). We use $k=10$ for these data sets. We further choose YearPredictionMSD (515345 points in $\mathbf{R}^{90}$)

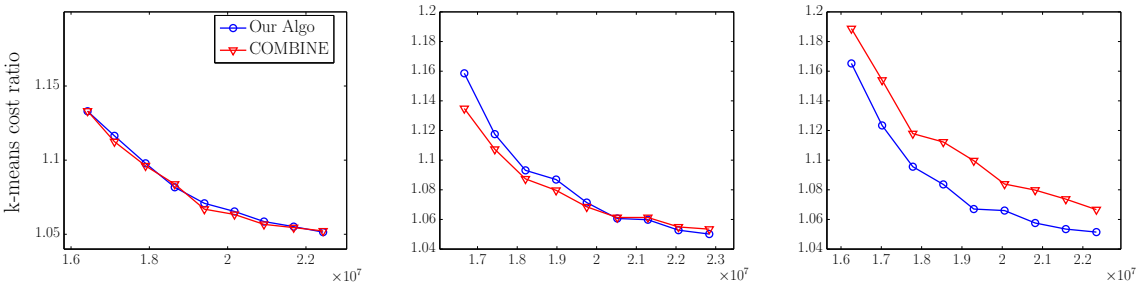
[5] Our theoretical analysis shows that our algorithm has better bounds on the communication cost. Since the bounds are from worst-case analysis, it is meaningful to verify that our algorithm also empirically outperforms other distributed coreset algorithms.

for larger scale experiments, and use $k = 50$ for this data set.

Experimental Methodology. To transform the centralized clustering data sets into distributed data sets we first generate a communication graph connecting local sites, and then partition the data into local data sets. To evaluate our algorithm, we consider several network topologies and partition methods.

The algorithms are evaluated on three types of communication graphs: random, grid, and preferential. The random graphs are Erdös-Renyi graphs $G(s, p)$ with $p = 0.3$, i.e. they are generated by including each potential edge independently with probability 0.3. The preferential graphs are generated according to the preferential attachment mechanism in the Barabási-Albert model [3]. For data sets Spam, Pendigits, and Letter, we use random/preferential graphs with 10 sites and 3×3 grid graphs. For synthetic data set and ColorHistogram, we use random/preferential graphs with 25 sites and 5×5 grid graphs. For large data set YearPredictionMSD, we use random/preferential graphs with 100 sites and 10×10 grid graphs.

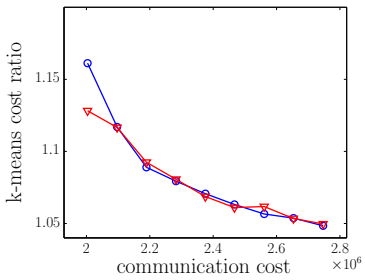
The data is then distributed over the local sites. When the communication network is a random graph, we consider three partition methods: uniform, similarity-based, and weighted. In the uniform partition, each data point in the global data set is assigned to the local sites with equal probability. In the similarity-based partition, each site has an associated data point randomly selected from the global data. Each data point in the global data is then assigned to the site with probability proportional to its similarity to the associated point of the site, where the similarities are computed by Gaussian kernel function. In the weighted partition, each local site is assigned a weight chosen by $|N(0, 1)|$ and then each data point is distributed to the local sites with probability proportional to the site's weight. When the network is a grid graph, we consider the similarity-based and weighted partitions. When the network is a preferential graph, we consider the degree-based partition, where each point is assigned with probability proportional to the site's degree.



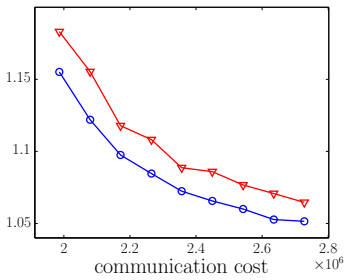
(a) random graph, uniform

(b) random graph, similarity-based

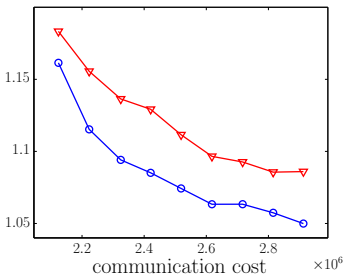
(c) random graph, weighted



(d) grid graph, similarity-based



(e) grid graph, weighted



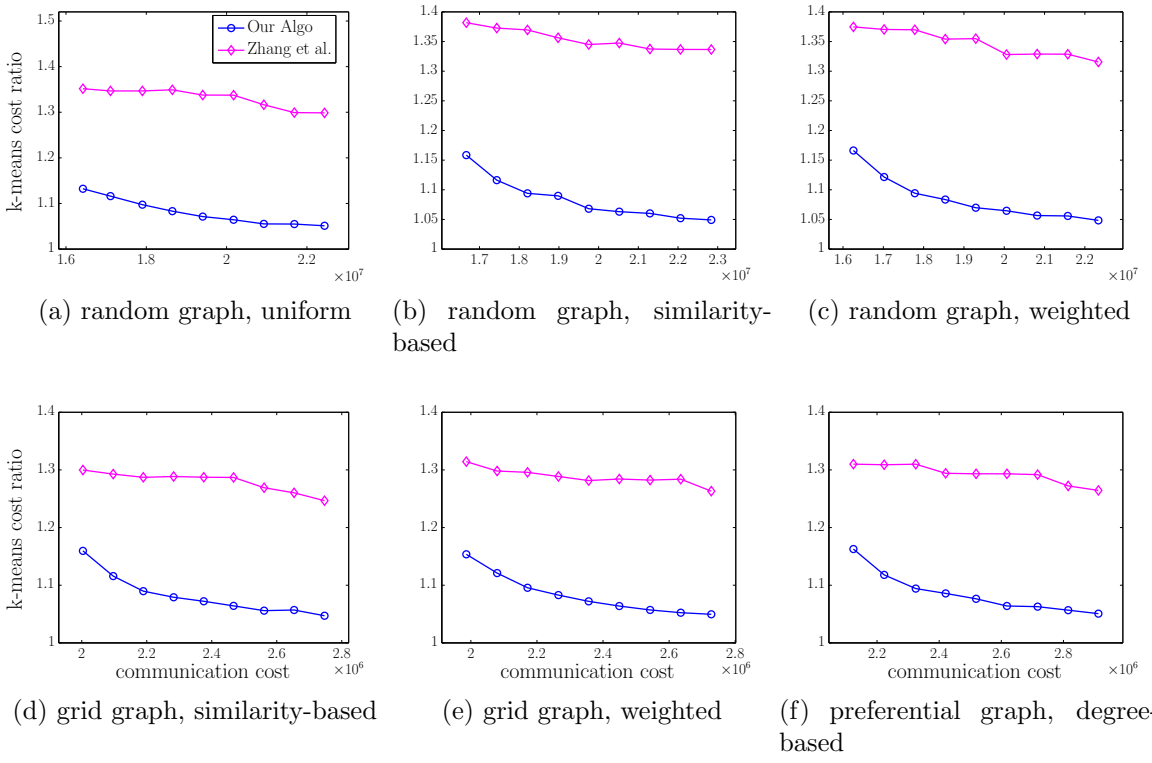
(f) preferential graph, degree-based

Figure 12: k-means cost (normalized by baseline) v.s. communication cost over graphs. The titles indicate the network topology and partition method.

To measure the quality of the coreset generated, we run Lloyd's algorithm on the coreset and the global data respectively to get two solutions, and compute the ratio between the costs of the two solutions over the global data. The average ratio over 30 runs is then reported. We compare our algorithm with COMBINE, the method of combining a coreset from each local data set, and with the algorithm of [104] (Zhang et al.). When running the algorithm of Zhang et al., we restrict the general communication network to a spanning tree by picking a root uniformly at random and performing a breadth first search.

Results. Here we focus on the results of the largest data set YearPredictionMSD, and in Appendix B.2 we present the experimental results for all the data sets.

Figure 14 shows the results over different network topologies and partition methods. We observe that the algorithms perform well with much smaller coreset sizes



(a) random graph, uniform

(b) random graph, similarity-based

(c) random graph, weighted

(d) grid graph, similarity-based

(e) grid graph, weighted

(f) preferential graph, degree-based

Figure 13: k-means cost (normalized by baseline) v.s. communication cost over the spanning trees of the graphs. The titles indicate the network topology and partition method.

than predicted by the theoretical bounds. For example, to get 1.1 cost ratio, the coreset size and thus the communication needed is only $0.1\% - 1\%$ of the theoretical bound.

In the uniform partition, our algorithm performs nearly the same as COMBINE. This is not surprising since our algorithm reduces to the COMBINE algorithm when each local site has the same cost and the two algorithms use the same amount of communication. In this case, since in our algorithm the sizes of the local samples are proportional to the costs of the local solutions, it samples the same number of points from each local data set. This is equivalent to the COMBINE algorithm with the same amount of communication. In the similarity-based partition, similar results are observed as it also leads to balanced local costs. However, when the local sites

have significantly different costs (as in the weighted and degree-based partitions), our algorithm outperforms COMBINE. As observed in Figure 14, the costs of our solutions consistently improve over those of COMBINE by $2\% - 5\%$. Our algorithm then saves $10\% - 20\%$ communication cost to achieve the same approximation ratio.

Figure 13 shows the results over the spanning trees of the graphs. Our algorithm performs much better than the algorithm of Zhang et al., achieving about 20% improvement in cost. This is due to the fact that their algorithm needs larger coresets to prevent the accumulation of errors when constructing coresets from component coresets, and thus needs higher communication cost to achieve the same approximation ratio.

Similar results are observed on the other data sets, which are presented in Appendix B.2.

3.4.2 Experiments on Distributed k-Means Clustering of High Dimensional Data

In these experiments we seek to understand how well the projected data approximates the original data, by measuring the k-means costs of the clustering solutions obtained after dimension reduction.

Data sets. We choose the following real world data sets from [11]: Daily and Sports Activities data (9210 points in $\mathbf{R}^{5625}$), MNIST handwritten digits ($70,000$ points in $\mathbf{R}^{784}$). We use $k = 10$ for these data sets. We further choose Bag of Words (NYTimes) ($300,000$ points in $\mathbf{R}^{102660}$) and use $k = 20$ for this data set.

Experimental Methodology. Following the setup in the last section, we first generate a communication graph, which can be a grid graph, or a random graph that includes each edge independently with probability 0.3. For Daily and Sports Activities data set, we use random graphs with 10 nodes and 3×3 grid graphs. For the other data sets, we use random graphs with 100 nodes and 10×10 grid graphs. Then we distribute the data over the graphs using weighted partition, where each

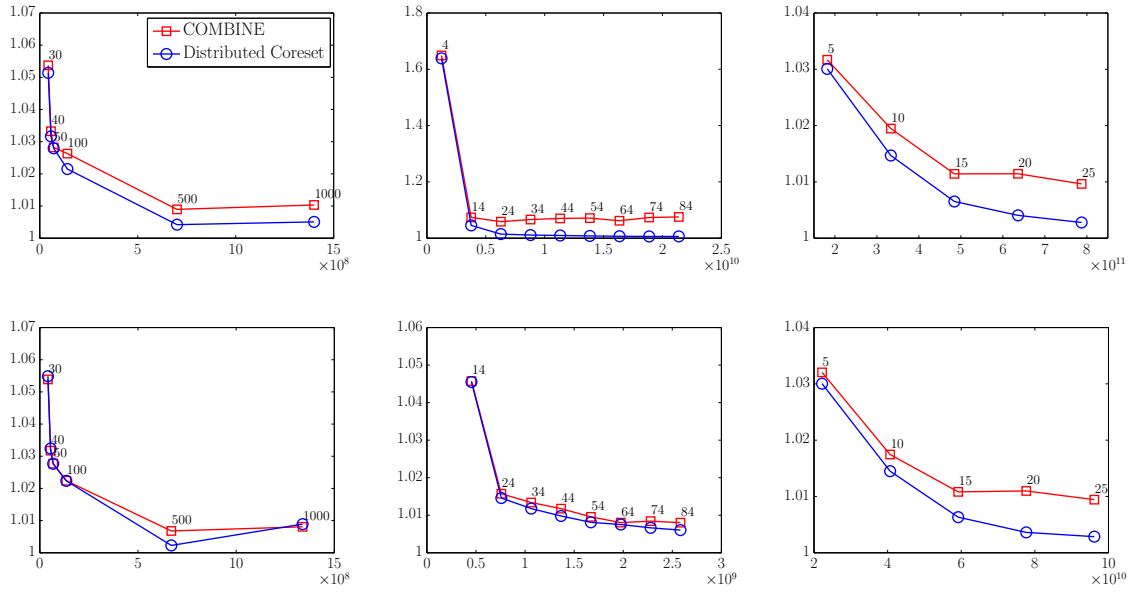


Figure 14: k-means cost (normalized by baseline) v.s. communication cost when the projection dimension varies. Rows: random graphs, and grid graphs. Columns: Daily and Sports Activities, MNIST, and Bag of Words data sets. In each subfigure, the x-axis represents the communication cost, the y-axis represents the k-means cost, and the number labels are the projection dimensions.

data point is distributed to the local sites with probability proportional to the site's weight chosen from $|N(0,1)|$.

For each projection dimension, we first construct a coreset on the projected data, using the COMBINE or distributed coreset algorithm in the last section. After building the coreset, we then run Lloyd's method on it to get a k-means clustering solution. Finally, we compute the ratio of its cost to the k-means cost obtained by running Lloyd's method on the original data. The average results over 10 runs are reported. We lower the projection dimension until there is a significant increase in the k-means costs.

Results. Figure 14 shows the results of the data sets. The plots show the increase in k-means cost ratio upon decreasing the dimension of the data. We can observe that there is a slight increase compared to the huge reduction in dimension and thus

communication cost. For example, on Daily and Sports Activities data, the k-means cost increases less than 4% when the dimension is reduced from 5625 to as low as 40. This is even more significant on higher dimensional data: on Bag of Words, the dimension can be reduced from 102660 to around 20. Such reduction then lowers the communication cost by magnitudes. The plots also indicate that the distributed coreset algorithm in the last section performs better than the COMBINE algorithm, when applied with our distributed PCA algorithm.

CHAPTER IV

COMMUNITY DETECTION

As we are entering into a network age, there is an increasing interest in analyzing network data in many disciplines, ranging from mathematics and computer science to sociology and biology. A significant amount of recent work in this area has focused on community detection, which is graph clustering viewing the network as a graph. The community structure reflects how entities in a network form meaningful groups such that interactions within the groups are more active compared to those between the groups and the outside world. The discovery of these communities is useful for understanding the structure of the underlying network, or making decisions in the network [49, 52, 88, 89].

Generally, a community should be thought of as a subset whose members have more interactions with each other than with the remainder of the network. This intuition is captured by some recently proposed models [34, 4, 21, 2, 58, 65]. Additionally, recent studies show that networks often exhibit hierarchical organization, in which communities can contain groups of sub-communities, and so forth over multiple scales. For example, this can be observed in ecological niches in food webs, modules in biochemical networks or groups of common interest in social websites [94, 71, 33]. It is also shown empirically and theoretically that hierarchical structures can simultaneously explain and quantitatively reproduce many commonly observed topological properties of networks [32, 97, 51]. This suggests that the hierarchical structure should also be reflected when modeling real world communities, which distinguishes community detection from the classic objective-based clustering approaches discussed in the previous chapters.

Although some heuristic approaches [51, 73] have been proposed to detect community hierarchies, few works have formalized this hierarchical property, and there are no theoretical performance guarantees for the algorithms. Inspired by the related work in clustering [16], in this chapter we define a notion of communities that reflects the tight connections within communities and explicitly models the hierarchy of communities. In our model, each member of a community falls into a sub-community, and the sub-communities within this community have active interactions with each other while entities outside this community have fewer interactions with members inside. Given this formalization, we then propose an efficient algorithm that detects all the communities in this model, and prove that all the communities form a hierarchy. Empirical evaluations demonstrate that our formalization successfully models real world communities, and our algorithm compares favorably with existing approaches.

4.1 Hierarchical Community Model

A network is typically represented as a graph $G = (V, E)$ on a set of $n = |V|$ points [1], where the edges could be undirected or directed, unweighted or weighted. The graph implicitly specifies a neighborhood structure on the points, i.e. for each point there is a ranking of all other points according to the level of possible interaction. More precisely, we assume that we have a neighborhood function N which given a point p and a threshold t outputs a list $N_t(p)$ containing the t nearest neighbors of p in V. Note that we do not assume the pairwise dissimilarity function for the points is a metric as in the previous chapters, but only assume the access to a neighborhood function.

The neighborhood function can be used to formalize a model of hierarchical communities. Using this neighborhood function, the tight connections within communities can be naturally rephrased as follows: for suitable t, most points p in the community

[1] We distinguish the nodes in the hierarchy our algorithm builds from the points in the graph.

have most of the nearest neighbors $N_t(p)$ from the community while points outside have just a few nearest neighbors from the community. Besides this, we also want to formalize the hierarchical structure that sub-communities in a lower, more local level actively interacting with each other form a community in a higher, more global level. The connections between the sub-communities can also be rephrased using the language of neighborhood: a majority of points in each sub-community have most of the nearest neighbors from the sub-communities in the same community.

In the remainder of the section, we specify our model based on the neighborhood function. We begin with the following notion of compact blobs, which will serve as a building block for our model.

Definition 12. *A subset A of points is called an α-compact blob, if out of the $|A|$ nearest neighbors:*

- *any point $p \in A$ has at most αn neighbors outside A, i.e. $|N_{|A|}(p) \setminus A| \leq \alpha n$;*
- *any point $q \notin A$ has at most αn neighbors inside A, i.e. $|N_{|A|}(q) \cap A| \leq \alpha n$.*

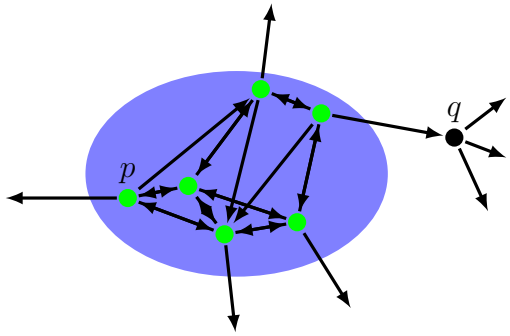


Figure 15: α-compact blob. An edge (x, y) means that y is one of x's nearest neighbors.

Note that the notion of compact blobs is the same as the clusters that satisfy the α-good neighborhood property defined in [16]. The notion captures the desired property of communities to be detected: members in the community have many more

interactions with other members inside the community and have fewer interactions with those outside. However, in practice, the notion may seem somewhat restricted. First, it requires all the members in the community have most interactions with other members inside the community, which may not be the case in real life. For example, some members in the boundary may have more interactions with the outside world, i.e. they have more than αn neighbors from outside. Based on this consideration, we define the (α, β)-stable property as follows.

Definition 13. *A community C is (α, β)-stable if*

- *any point $p \in C$ falls into a α-compact blob $A_p \subseteq C$ of size greater than $6\alpha n$,*
- *for any point $p \in C$, at least β fraction of points in A_p have all but at most αn nearest neighbors from C out of their $|C|$ nearest neighbors,*
- *any point q outside C has at most αn nearest neighbors from C out of their $|C|$ nearest neighbors.*

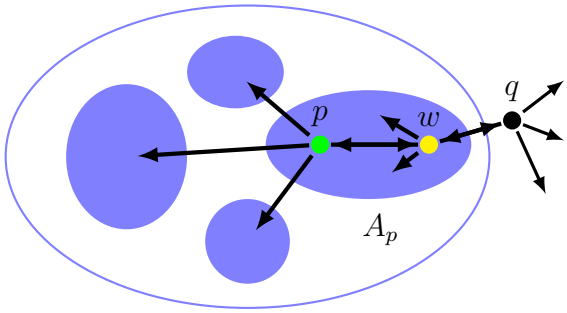


Figure 16: (α, β)-stable community. An edge (x, y) means that y is one of x's nearest neighbors. Note that point w lies on the "boundary" of the community. It falls into the compact blob A_p, but does not have most of its nearest neighbors from the community.

Informally, the first condition means that every point falls into a sufficiently large compact blob in its community. This condition formalizes the local neighborhood structure that each member interacts actively with sufficiently many members in

the community. Note that the compact blob should be large enough so that the membership of the point is clearly established. That is, it should have size comparable to αn, the number of connections to points outside. Here we choose a minimum size of $6\alpha n$ mainly because it guarantees that our algorithm can still identify the blob in the worst case. The second condition means that at least β fraction of points in these compact blobs have most of their nearest neighbors from the community. This condition formalizes more global neighborhood structure about how the compact blobs interact with each other to form a community. The third condition formalizes how the community is separated from the outside.

Note that we no longer require all the members in the community have most interactions inside; we only require each member interacts with sufficiently many members and a majority of members in these local groups interact actively. Also note that the definition is hierarchical in nature: sufficiently large compact blobs clearly satisfy the definition of (α, β)-stable property and thus can be viewed as communities in lower levels. Furthermore, in the next section we will show that all the (α, β)-stable communities form a hierarchy. We show this by presenting an algorithm and proving that each (α, β)-stable community is a node in the hierarchy output by the algorithm. So our formulation explicitly models the hierarchical structure of communities observed in networks.

Next we propose a further generalization that considers possible noise in real world data. There may be some abnormal points that do not exhibit clear membership to any community, in the presence of which our definition above does not model the communities well. For example, suppose there is a point that has connections to all other points in the network, then no non-trivial subsets satisfy our definition above. We call such points bad since they do not fit into our community model above. To deal with the noise, we can naturally relax the (α, β)-stable property to the (α, β, ν)-stable property defined as follows. Informally, it requires that the target community satisfies

the (α, β)-stable property after removing a few bad points B. For convenience, we call the other points in $V \setminus B$ good points.

Definition 14. *A community C is (α, β, ν)-stable if there exist a subset of bad points B of size at most νn, such that*

- *any good point $p \in G = C \setminus B$ falls into a compact blob $A_p \subseteq C$ of size greater than $6(\alpha + \nu)n$,*
- *for any point $p \in G$, at least β fraction of points in A_p have all but at most αn nearest neighbors from G out of their $|G|$ nearest neighbors in $V \setminus B$,*
- *any good point q outside $C \cup B$ has at most αn nearest neighbors from G out of their $|G|$ nearest neighbors in $V \setminus B$.*

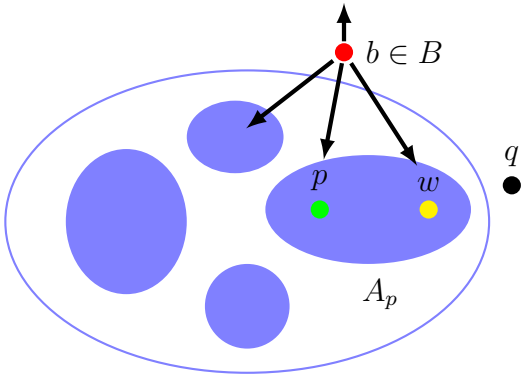


Figure 17: (α, β, ν)-stable community. An edge (x, y) means that y is one of x's nearest neighbors. Point b is a bad point and does not exhibit clear membership to any community.

Note 1 The parameters α, ν are defined globally, that is, they are defined as ratios with respect to the total number of points. So a local change to some community can affect the values of these parameters for the other communities. For example, suppose we add Kn new points to some community, with all the new points having neighbors only inside this special community. Since the number of points increases

to $(K+1)n$, the communities outside the modified community are now $(\alpha/(1+K), \beta, \nu/(1+K))$-stable. However, the local change does not affect the identifiability of these communities. Our algorithm described in the next section can still detect these communities, given the value of $(\alpha+\nu)n$.

Note 2 The input of the community detection task is usually a graph representing the network, and there are different ways to lift the graph to a neighborhood function. The simplest one is to directly sort for each point p all the other points q according to the weights of the edges (p, q) and break ties randomly (we assume without loss of generality that the weights are in $[0, 1]$ and the weight of an edge not in E is regarded as 0). However, as pointed out in [21], we also have alternative approaches to convert the observed graph into a neighborhood function. More specifically, we assume the observed graph reflects some underlying unobserved set of relations, and thus we can lift the graph to an affinity system based on various beliefs about the connection between the latent relations and the observed graph, and then sort the points according to the affinity system to get the neighborhood function. For example, based on the belief that random walks on the graph can reflect the similarities between entities, we can define the affinity to be the diffusion kernel $\exp\{\lambda A\}$ where A is the adjacent matrix and λ is a parameter. Note that the results of appropriate lifting procedures can better reflect the true relationships between entities, and thus the conversion can address the challenging issue of sparsity in the observed graph.

4.2 Hierarchical Community Detection Algorithm

In the section, we propose an algorithm for detecting communities satisfying the (α, β, ν)-stable property. The goal of our algorithm is to output a set of communities such that each community satisfying the (α, β, ν)-stable property is close to one in the output. To be precise, we say that a community C is ν-close to another community C' if $|C \setminus C'| + |C' \setminus C| \leq \nu n$. We first describe the details in Algorithm 12, and then

Algorithm 12 Hierarchical Community Detection Algorithm

Input: Neighborhood function N on a set of points V, $n = |V|$, $\alpha > 0, \nu > 0$.

Initialize $\mathcal{C}'$ to be a set of singleton points, and $t = 6(\alpha + \nu)n + 1$.
while $|\mathcal{C}'| > 1$ **do**
  Build F_t on V as follows.
  for any $x, y \in V$ that satisfy $|N_t(x) \cap N_t(y)| \geq t - 2(\alpha + \nu)n$ **do**
    Connect x, y in F_t.
  end for
  Build H_t on $\mathcal{C}'$ as follows. Let $N_F(x)$ denote the neighbors of x in F_t.
  for any $U, W \in \mathcal{C}'$ **do**
    if U, W are singleton subsets, i.e. $U = \{x\}, W = \{y\}$ **then**
      Connect U, W in H_t, if $|N_F(x) \cap N_F(y)| > (\alpha + \nu)n$.
    else
      Set $S_t(x, y) = |N_F(x) \cap N_F(y) \cap (U \cup W)|$, $\forall x \in U, y \in W$.
      Connect U, W in H_t, if $\text{median}_{x \in U, y \in W} S_t(x, y) > \frac{|U|+|W|}{4}$.
    end if
  end for
  for any component R in H_t that satisfies $|\cup_{C \in R} C| \geq 4(\alpha + \nu)n$ **do**
    Update $\mathcal{C}'$ by merging subsets in R into one subset.
  end for
  $t = t + 1$.
end while

Output: Hierarchy T with single points as leaves and internal nodes corresponding to the merges performed.

present the analysis in Theorem 16.

Now we prove that the algorithm successfully outputs a hierarchy such that any community satisfying the (α, β, ν)-stable property with sufficiently large β is close to one of the nodes in the hierarchy. Formally,

Theorem 16. *Algorithm 12 outputs a hierarchy such that any community satisfying the (α, β, ν)-stable property with $\beta \geq 5/6$ is ν-close to a node in the hierarchy. The algorithm runs in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

The correctness of the theorem follows from Lemma 21 and the running time follows from Lemma 22. In the following analysis, we always assume $\beta \geq 5/6$. Before presenting the analysis for the general communities in Lemma 21, we first prove a

lemma for the base case of compact blobs, showing that for any compact blob, a node close to it will be formed.

Lemma 19. *For any good point p, when $t \leq |A_p|$, good points from A_p will not be merged with good points outside A_p. At the end of the threshold $t = |A_p|$, all points in A_p have been merged into a subset.*

Proof. We prove this by induction on t. The claim is clearly true initially. Now assume for induction that at the beginning of a threshold $t \leq |A_p|$, in $\mathcal{C}'$ good points from A_p are not merged with good points outside A_p, i.e. any subset can contain good points from only one of A_p and $V \setminus B \setminus A_p$. We now analyze the properties of the graphs F_t and H_t, and show that at the end of the current threshold, the claim is still true.

First, as long as $t \leq |A_p|$, the graph F_t has the following properties.

- No good point x in A_p is connected to a good point y outside A_p. By the definition of compact blobs, out of the t nearest neighbors, x has at most $(\alpha+\nu)n$ neighbors outside A_p. For $y \in V \setminus B \setminus A_p$, y has at most $(\alpha + \nu)n$ neighbors in A_p. Then x, y have at most $2(\alpha + \nu)n < t - 2(\alpha + \nu)n$ common neighbors, so they are not connected.

- No bad point z is connected to both a good point x in A_p and a good point y outside A_p. We know that out of the t nearest neighbors, x has at most $(\alpha+\nu)n$ neighbors outside A_p. So if z is connected to x, then z must have more than $t - 3(\alpha + \nu)n$ neighbors in A_p and less than $3(\alpha + \nu)n$ neighbors outside A_p. Since y has at most $(\alpha + \nu)n$ neighbors in A_p, we have that y, z share less than $3(\alpha + \nu)n + (\alpha + \nu)n < t - 2(\alpha + \nu)n$ neighbors, so they are not connected.

Based on the properties of F_t and the inductive assumption that any subset can contain good points from only one of A_p and $V \setminus B \setminus A_p$, we show that the graph H_t has the following properties.

- No subset U containing good points from A_p is connected to a subset W containing good points outside A_p. This is clearly true if they are singleton subsets. In the other cases, note that the fraction of bad points in U or W is at most $1/4$. Then the number of pairs (x, y) with good points $x \in U$ and $y \in W$ is at least $\frac{3}{4}|U| \times \frac{3}{4}|W| > |U||W|/2$, i.e. more than half of the pairs (x, y) with $x \in U$ and $y \in W$ are pairs of good points. This means there exist good points $x^* \in U, y^* \in W$ such that $S_t(x^*, y^*)$ is no less than $\text{median}_{x \in U, y \in W} S_t(x, y)$. By the properties of F_t, x^*, y^* have no common neighbors. Therefore, U and W are not connected.

- If a subset W contains only bad points, then it cannot be connected to both a subset containing good points from A_p and a subset containing good points outside A_p. Suppose it is connected to U which contains good points from A_p. Note that since W contains only bad points, it must contain only a single point z. If $U = \{x\}$ is singleton, then x, z share more than $(\alpha + \nu)n$ neighbors in F_t. Since in F_t, x is only connected to good points from A_p and bad points, z and x must share some common neighbors from A_p, then z must be connected to some good points in A_p. In the other cases, note that the fraction of bad points in U is at most $1/4$. So there exists a good point $x^* \in U$ such that $S_t(x^*, z) \geq \text{median}_{x \in U} S_t(x, z)$. Then we have $S_t(x^*, z) > (|U| + |W|)/4 > \nu n$, and thus z must also be connected to some good points in A_p. Similarly, if W is connected to a subset containing good points outside A_p, then the point in W must connect to some good point outside A_p. But this is contradictory to the fact that in F_t no bad point is connected to both a good point in A_p and a good point outside A_p.

By the properties of H_t, no connected component contains both good points in A_p and good points outside A_p. So at the end of this threshold t, the claim is still

true. Then by induction, we know that when $t \leq |A_p|$, we will not merge good points from A_p with good points outside A_p.

Next we show that at the end of the threshold $t = |A_p|$, we will merge all points in A_p into a subset. First, at this threshold, all good points in A_p are connected in F_t. Any good point in A_p has at most $(\alpha+\nu)n$ neighbors outside A_p, so when $t = |A_p|$, any two good points x, y in A_p are connected, and thus they share at least $|A_p|$ common neighbors in F_t. Second, all subsets containing good points in A_p are connected in H_t. If no good points in A_p have been merged, then these singleton points will be connected in H_t since they share at least $|A_p|$ singleton subsets as common neighbors in F_t. If some good points in A_p have already been merged into non-singleton subsets, we can show that in H_t these non-singleton subsets will be connected to each other and connected to singleton subsets containing good points from A_p. For any such pair of subsets U and W, the fraction of bad points in U or W is at most $1/4$, so there exist good points $x^* \in U, y^* \in W$ such that $\text{median}_{x\in U, y\in W} S_t(x, y)$ is no less than $S_t(x^*, y^*)$. Since x^*, y^* are connected to all good points in A_p in F_t, $S_t(x^*, y^*)$ is no less than the number of good points in U and W. So $\text{median}_{x\in U, y\in W} S_t(x, y) \geq S_t(x^*, y^*) > (|U| + |W|)/4$, and thus U, W are connected in H_t. Therefore, all points in A_p are merged into a subset. □ □

The following is a consequence of Lemma 19, which will be used in the analysis for the general communities in Lemma 21.

Lemma 20. *In Algorithm 12, if a subset U satisfies that for any good point $p \in U$, $A_p \subseteq U$, then there exist a subset of good points $P \subseteq U$, such that $\{A_p : p \in P\}$ is a partition of $U \setminus B$.*

Proof. We have $U \setminus B = \cup_{p\in U\setminus B} A_p$. We only need to show that sets in $\{A_p : p \in U \setminus B\}$ are laminar, i.e. for any $p, q \in U \setminus B$, either $A_p \cap A_q = \emptyset$ or $A_p \subseteq A_q$ or $A_q \subseteq A_p$. Assume for contradiction that there exist A_p and A_q such that $A_p \setminus A_q \neq \emptyset, A_q \setminus A_p \neq \emptyset$

and $A_p \cap A_q \neq \emptyset$. Without loss of generality, suppose $|A_p| \leq |A_q|$. Then by Lemma 19, at the end of the threshold $t = |A_p|$, we have merged all good points in A_p into a subset. Specifically, this means that we have merged $A_p \cap A_q$ with $A_p \setminus A_q$. So for $t \leq |A_q|$, we have merged good points in A_q with good points outside A_q, which is contradictory to Lemma 19. □ □

By the above lemmas, for any good point p, the subset A_p will be formed before points in it are merged with good points outside. Once these subsets are formed, we can show that subsets in the same target community will be merged together before they are merged with those from other communities, and thus the hierarchy produced has a node close to the target community. Formally, we have the following result.

Lemma 21. *For any community C satisfying the (α, β, ν)-stable property with $\beta \geq 5/6$, $\mathcal{C}' \setminus B$ in Algorithm 12 is always laminar to $C \setminus B$, i.e. for any $C' \in \mathcal{C}'$, either $(C' \setminus B) \cap (C \setminus B) = \emptyset$ or $(C' \setminus B) \subseteq (C \setminus B)$ or $(C \setminus B) \subseteq (C' \setminus B)$. Furthermore, there is a node U in the hierarchy produced such that $U \setminus B = C \setminus B$.*

Proof. we will show by induction on t that: for any community C satisfying the (α, β, ν)-stable property with $\beta \geq 5/6$,

- at the end of threshold t, $\mathcal{C}' \setminus B$ is laminar to $C \setminus B$,
- at the end of threshold t, for any C such that $|C \setminus B| \leq t$, we have merged all points in $C \setminus B$ into a subset.

These claims are clearly true initially. Assume for induction that they are true for the threshold $t - 1$, we now show that they are also true for the threshold t.

We first show that the laminarity is preserved. The laminarity is broken only when we connect in H_t two subsets U, W such that U is a strict subset of C after removing the bad points, and W is a subset containing good points from outside. If there is a good point $p \in U$ such that $A_p \not\subseteq U$, then by Lemma 19, they cannot be connected.

So we only need to consider the other case when for any good point $p \in U, A_p \subseteq U$. For convenience, we call a point great if it is a good point in C, and it has less than αn neighbors outside $C \setminus B$ out of the $|C \setminus B|$ nearest neighbors in $V \setminus B$. We now show that U, W are not connected in H_t. Since $U \setminus B$ is a strict subset of $C \setminus B$, by induction on the second claim, we have $t \leq |C \setminus B|$. Then great points in U and points in W share at most $2(\alpha+\nu)n < t - 2(\alpha+\nu)n$ common neighbors, so they are not connected in F_t. By Lemma 20 and the second condition of the (α, β, ν)-stable property, we know that at least $5/6$ fraction of points in $U \setminus B$ are great points. Then there exist a great point $x^* \in U$ and a point $y^* \in W$ such that $S_t(x^*, y^*)$ is no less than $\text{median}_{x \in U, y \in W} S_t(x, y)$. Since in F_t great points in U are not connected to points in W, we have $S_t(x^*, y^*) \leq (|U| + |W|)/4$. So $\text{median}_{x \in U, y \in W} S_t(x, y) \leq (|U| + |W|)/4$ and U, W are not connected in H_t. Therefore, the laminarity is preserved.

Next we show that at the end of the threshold $t = |C \setminus B|$, all points in $C \setminus B$ are merged into a subset. By Lemma 19, all good points in $C \setminus B$ are now in sufficiently large subsets. We claim that any two of these subsets U, W are connected in H_t, and thus will be merged. Again by Lemma 20, we know at least $5/6$ fraction of points in $U \setminus B$ or $W \setminus B$ are great points, and thus there exist great points $x^* \in U, y^* \in W$ such that $S_t(x^*, y^*)$ is no more than $\text{median}_{x \in U, y \in W} S_t(x, y)$. Notice that all great points in U are connected to great points in W in F_t, since they share at least $t - 2(\alpha+\nu)n$ neighbors. Then $S_t(x^*, y^*) \geq 3(|U| + |W|)/4 > (|U| + |W|)/4$, and thus $\text{median}_{x \in U, y \in W} S_t(x, y) > (|U| + |W|)/4$. Therefore, any two subsets containing good points from $C \setminus B$ are connected in H_t and thus are merged.

So the two claims hold for all t, specially for $t = n$. Then the algorithm must stop after this threshold, and we have the lemma as desired. □ □

Lemma 22. *Algorithm 12 has a running time of $O(n^{\omega+1})$.*

Proof. To implement the algorithm, we introduce some data structures. For any $x \in V$, if y is within the t nearest neighbors of x, let $I_t(x, y) = 1$, otherwise $I_t(x, y) =$

0. Initializing I_t takes $O(n^2)$ time. Next we compute $CN_t(x, y)$, the number of common neighbors between x and y. Notice that $CN_t(x, y) = \sum_{z \in V} I_t(x, z) I_t(y, z)$, so $CN_t = I_t I_t^T$. Then we can compute the adjacent matrix F_t (overloading notation for the graph F_t) from CN_t. These take $O(n^\omega)$ time.

To compute the graph H_t, we introduce the following data structures. Let $FS_t(x, y) = 1$ if x, y are singleton subsets and $F_t(x, y) = 1$, and let $FS_t(x, y) = 0$ otherwise. Let $NS_t = FS_t(FS_t)^T$, then for two singleton subsets x, y, $NS_t(x, y)$ is the number of singleton subsets they share as neighbors in common in F_t. Similarly, let $FC_t(x, y) = 1$ if x and y are in the same subset and $F_t(x, y) = 1$, and let $FC_t(x, y) = 0$ otherwise. Let $S_t(x, y) = NS_t(FC_t)^T + FC_t(NS_t)^T$, then for two points $x \in U, y \in W$ where U, W are two non-singleton subsets, $S_t(x, y)$ is the number of points in $U \cup W$ they share as neighbors in common in F_t. Based on NS_t and S_t we can build the graph H_t. All these take $O(n^\omega)$ time.

When we perform merge or increase the threshold, we need to update the data structures, which takes $O(n^\omega)$ time. Since there are $O(n)$ merges and $O(n)$ thresholds, Algorithm 12 takes time $O(n^{\omega+1})$ in total. □ □

4.3 Local Algorithm for Community Detection

For networks with millions of nodes, the algorithm in the last section is too computationally expensive. Fortunately, in practical networks, the communities often have size much smaller than that of the whole network. For example, a social network can have millions members, while a member typically has no more than hundreds of close friends. Therefore, it is often sufficient to consider the following problem:

- Given some p, which is a good point in a community C, and the size of C, how to identify C in time independent of the size of the whole network?

Such algorithms are called local algorithms in the literature of community detection. Note that the independence of the size of the whole network can only be achieved

Algorithm 13 Local Community Detection Algorithm

Input: Neighborhood function N on a set of points V, a point $p \in V$, an integer $c > 0$, parameters $\alpha > 0, \nu > 0$.

1: Let $N_t(A)$ denote the union of the t nearest neighbors of points in A. Set $N^2 = N_{2c}(N_{2c}(p))$.
2: Set $C'_{\text{in}} = \{q \in N^2 : |N_c(q) \cap N_{6(\alpha+\nu)n}(p)| \geq 3(\alpha+\nu)n\}$.
3: Set $C' = \{q \in N^2 : |N_c(q) \cap C'_{\text{in}}| \geq 3(\alpha+\nu)n\}$.

Output: C'.

under some assumption about the input, since reading the similarities between p and the other points already takes time linear in the size of the network. Here we assume that the algorithm has access to a sorted neighborhood function N_t. Formally, we assume the following mild condition.

(**A**) For any p and t, $N_t(p)$ can be computed in time $O(t)$.

This assumption can be satisfied by building a sorting list of nearest neighbors for each point, which can be precomputed once in time $O(n^2 \log n)$ from the pairwise similarities.

Here we design a local algorithm (described in Algorithm 13), which solves the aforementioned problem under Assumption (**A**). To analyze the algorithm, first recall the definition of the stable community C, which requires some good points in the community have all but at most αn nearest neighbors from $C \setminus B$ out of their $C \setminus B$ nearest neighbors in $V \setminus B$. Call them inner points of C and denote as C_{in} for convenience. Intuitively, the algorithm is designed such that C'_{in} is a good estimation of C_{in}. From this, the algorithm then builds C' such that $C' \approx C$. The construction is localized to an approximate superset N^2 of C, so that the running time is independent of the size of the network.

Theorem 17. *Suppose C is an (α, β, ν)-stable community with $\beta \geq 5/6$. Given $p \in C \setminus B$ and $c = |C|$, Algorithm 13 outputs C' such that $|C' \setminus C| + |C \setminus C'| \leq (\alpha+\nu)n$. The running time is $O(|C|^4)$ under Assumption (**A**).*

Proof. First, N^2 is approximately a superset of C. More precisely, we have $|C \setminus B \setminus N^2| \leq \alpha n$. By definition, at most αn points in A_p are outside $N_{2c}(p)$. Since at least $\beta \geq 5/6$ fraction of A_p are inner points, $N_{2c}(p)$ contains at least one inner point. Then $N^2 = N_{2c}(N_{2c}(p))$ contains all but at most αn good points in C. For analysis, denote these missing points as $M = C \setminus B \setminus N^2$.

Second, C'_{in} contains all inner points in $C \setminus M$ but contains no good points outside C. By definition, $|A_p \cap N_{6(\alpha+\nu)n}(p)| \geq 5(\alpha+\nu)n$. If $q \in C_{\text{in}}$, then $|A_p \setminus N_c(q)| \leq (\alpha+\nu)n$. Combining the two leads to $|N_c(q) \cap N_{6(\alpha+\nu)n}(p)| > 3(\alpha+\nu)n$ since A_p has size at least $6(\alpha+\nu)n$. This means that C'_{in} contains all inner points in $C \setminus M$. To show that C'_{in} contains no good points q outside C, note that $|N_c(q) \cap A_p| \leq (\alpha+\nu)n$. Also, $|N_{6(\alpha+\nu)n}(p) \setminus A_p| \leq (\alpha+\nu)n$. So we have $|N_c(q) \cap N_{6(\alpha+\nu)n}(p)| \leq 2(\alpha+\nu)n$.

Finally, C' contains all good points in $C \setminus M$ but contains no good points outside C, which then means $|C' \setminus C| + |C \setminus C'| \leq (\alpha+\nu)n$. If $q \in C \setminus B$, then $|A_q \setminus N_c(q)| \leq (\alpha+\nu)n$. Since C'_{in} contains all points in $C_{\text{in}} \setminus M$, and at least β fraction of the points A_q are in C_{in}, we have that $|A_q \cap C'_{\text{in}}| \geq \beta|A_q| - |M| \geq 4(\alpha+\nu)n$. Then $|C'_{\text{in}} \cap N_c(q)| \geq 3(\alpha+\nu)n$, and thus C' contains all good points in $C \setminus M$. To show C' contains no good points q outside C, note that $|N_c(q) \cap C| \leq (\alpha+\nu)n$. Since $C'_{\text{in}} \subseteq C \cup B$, we have $|N_c(q) \cap C'_{\text{in}}| \leq (\alpha+\nu)n + |B| < 3(\alpha+\nu)n$.

To bound the running time, first note that N^2 can be computed in time $|C|^2$ under Assumption ((A)). To build C'_{in}, we can search over all $O(|C|^2)$ points in N^2. For each $q \in N^2$, finding $N_c(q)$ takes time $O(|C|)$; for each $q' \in N_c(q)$, testing whether $q' \in N_{6(\alpha+\nu)n(p)}$ takes time $O((\alpha+\nu)n) = O(|C|)$. Then the total time to build C'_{in} is $O(|C|^4)$. Similarly, building C' takes time $O(|C|^4)$. Therefore, the algorithm takes time $O(|C|^4)$. □

4.4 Experiments

In this section, we present our experimental results on evaluating our model and algorithm. While our main concern is building theoretical model for communities, empirical study is valuable in verifying the model and providing guidance for further improvement. Therefore, we applied our algorithm on both real world and synthetic data sets.

Note that the networks are represented as graphs, and we need to lift the graphs to get neighborhood functions for our algorithm. We use two lifting approaches for our experiments. The first approach is direct lifting: first, for any x, y set the affinity between x and y to be 1 if $(x, y) \in E$ and 0 otherwise; then for each x, sort all the other points according to the affinities; break ties randomly to avoid bias. The second approach is diffusion lifting: first set the affinity matrix K between entities to be $K = \exp\{\lambda A\}$ where $\lambda = 0.05$ and A is the adjacent matrix of the graph; then for each x, sort all the other points according to the affinities.

For comparison, we implemented two other algorithms: the lazy random walk algorithm (LRW [99]) and the Girvan-Newman algorithm (GN [51]). The lazy random walk algorithm performs truncated random walk from a seed point in the network and outputs selected communities where the selection is guided by the walk distribution and conductance. The conductance has been widely used as a criterion for quantifying the tight connections within communities, and thus the comparison to the lazy random walk algorithm provides an evaluation on how well our model and algorithm capture this intuition. The GN algorithm repeatedly removes the edge with the maximum edge-betweenness and regards the created connected components as communities. Although no theoretical model of hierarchical communities is targeted, the algorithm builds a hierarchy during its execution. It has been shown that the algorithm performs remarkably well on modeling communities in real-world data sets [51, 89]. We use the code from [28] for fast computation of edge betweenness in

the algorithm.

For algorithms with parameters, we run them multiple times with different values of parameters, and report the best result. More specifically, we run our algorithm using parameters $(\alpha + \nu) = \frac{i}{5n}(i = 1, 2, \ldots, 5)$. For the lazy random walk algorithm, we enumerate the parameters $\theta_0 = 0.05i(i = 1, \ldots, 4)$ and $b = 1, 2, \ldots, \lceil \log m \rceil$. In each run, 100 seed points are generated uniformly at random, each of which leads to a community. Since not all communities are meaningful (e.g. a singleton subset or the entire set of points), communities containing less than 10 points or containing more than $n - 10$ points are removed, and the rest communities are regarded as the output communities. We then evaluate the average error of the output communities. The error for a ground-truth community C with respect to a set $\mathcal{C}$ of output communities is defined as

$$\text{error}(C, \mathcal{C}) = \min_{C' \in \mathcal{C}} \frac{|C \setminus C'| + |C' \setminus C|}{n}.$$

This criterion measures how well the ground-truth communities are recovered by the algorithm. We further note that our algorithm outputs fewer communities than the other algorithms in all the conducted experiments, and thus has advantage when they achieve similar performance.

4.4.1 Evaluation on Real-world Networks

To assess the performance of the proposed method in terms of accuracy, we conduct experiments on the following real world data sets [2] : karate [103], dolphins [82], polbooks [68], and football [51].

Figure 18 shows the average error and running time of the algorithms. We observe that our algorithm with diffusion lifting achieves the best performance on 3 out of 4 data sets, and achieves performance comparable to the GN algorithm on the football data set. It recovers the ground truth communities remarkably well over all the data

[2] Detailed descriptions and links for download can be found on `http://www-personal.umich.edu/~mejn/netdata/`.

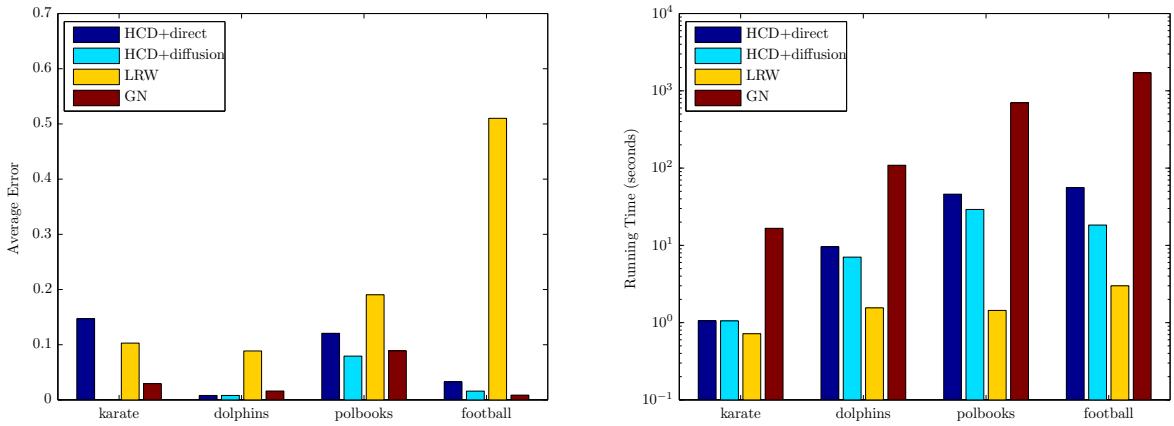


Figure 18: The average error and running time using our hierarchical community detection algorithm with direct lifting (HCD+direct) or diffusion lifting neighborhood function (HCD+diffusion), Lazy Random Walk (LRW [99]) and the Girvan-Newman algorithm (GN [51]). Note that the running time is in log scale.

sets. Our algorithm with direct lifting does not achieve good results. Note that this is due to the fact that diffusion lifting reflects the true neighborhood structure more accurately than direct lifting. More precisely, when we sort neighbors for a point p in direct lifting, all points not adjacent to p are ranked randomly. In fact some of them can be reached by a few steps and thus should be ranked as close neighbors, while others are actually far away from the point p. On the other hand, diffusion lifting leads to a neighborhood function that more accurately reflects the neighborhood information. The LRW algorithm has the worst performance, though it is the fastest. Our algorithm, especially with the diffusion lifting, runs 10-100 faster than the GN algorithm. Therefore, our algorithm with suitable neighborhood functions is the most favorable for detecting real world communities.

Data set	n	m	k	$maxk$	$minc$	$maxc$	$minC$	$maxC$
LF50	50	≈500	10	15	10	15	20	30
LF100	100	≈1500	15	20	15	20	30	40
LF150	150	≈3000	20	30	20	30	40	60
LF200	200	≈6000	30	40	30	40	60	80

Table 1: The parameters of the synthetic data sets for community detection.

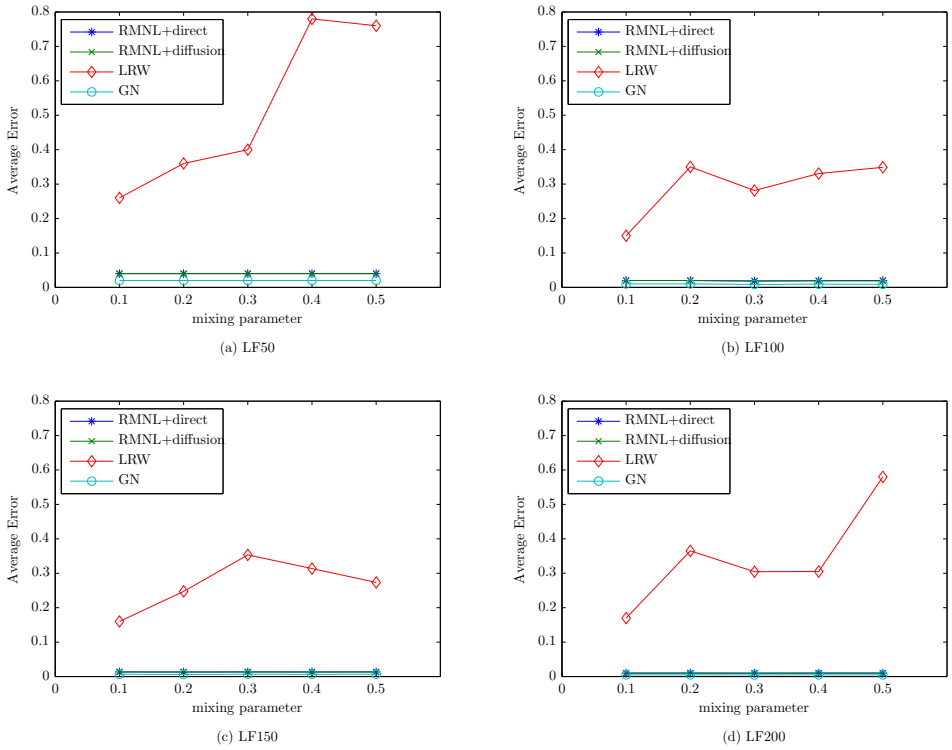


Figure 19: The average error on the synthetic data sets

4.4.2 Evaluation on Synthetic Networks

Besides real-world networks, we further use the Lancichinetti-Fortunato (LF) benchmark [3] graphs [72] to evaluate the performance of the algorithms. By varying the parameters of the networks, we can analyze the behavior of the algorithms in detail.

[3]The source code we use and details about the parameters can be found on `https://sites.google.com/site/andrealancichinetti/software`.

We generate four unweighted undirected benchmark networks with two level community hierarchies. The numbers of nodes are $50, 100, 150$ and 200 respectively, and some important parameters of the networks are given in Table 1. The parameters are as follows: n/m: number of nodes/edges; $k/maxk$: average/maximum degree of the nodes; $minc/maxc$: minimum/maximum size of the lower level communities; $minC/maxC$: minimum/maximum size of the higher level communities. For each type of data set, we range the mixing parameter μ from 0.1 to 0.5 with a span of 0.1, and set the low-level mixing parameter $\mu_1 = \mu/4$ and the high-level mixing parameter $\mu_2 = \mu - \mu_1$, resulting in five networks. Generally, the higher the mixing parameter of a network is, the more difficult it is to reveal the community structure.

Figure 19 shows the average errors of the algorithms and Figure 20 shows the running time. Our algorithm with direct or diffusion lifting and the GN algorithm achieve similar results on all the benchmark networks. The errors of these algorithms are below 5%, and hardly increase with the mixing parameter. This suggests that they recover the ground truth communities remarkably well even in the hard case when the members of the communities have significant connections with the outside. In contrast, the LRW algorithm does not recover the communities well, even though it runs much faster than the other algorithms. Our algorithm runs about 50 times faster than the GN algorithm over all the data sets. These results are consistent with those observed on real world data sets, and again demonstrate the advantage of our algorithm.

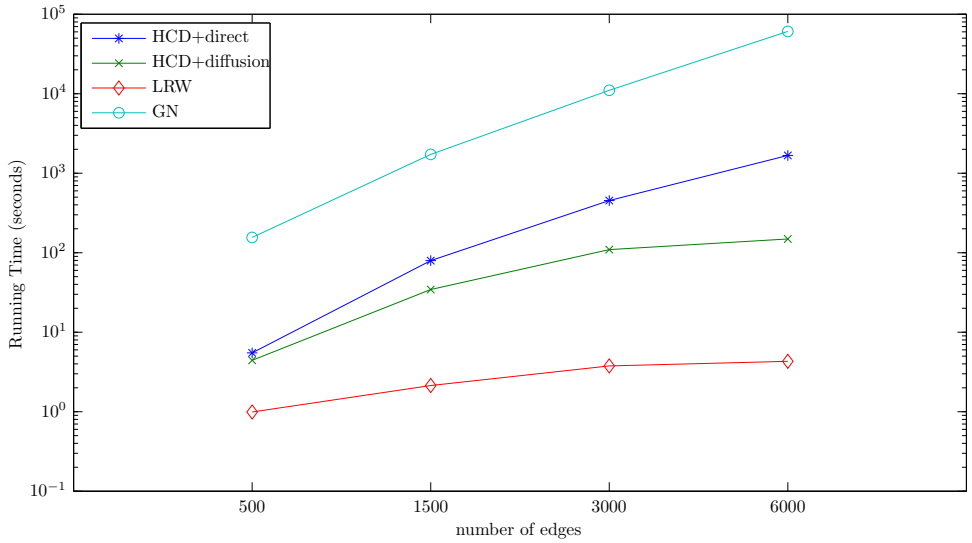


Figure 20: The running time on the synthetic data sets

APPENDIX A

CLUSTERING UNDER PERTURBATION RESILIENCE

A.1 Dynamic Programming to Find the Minimum Cost k-Cluster Pruning

The idea of using dynamic programming to find the optimal k-clustering in a tree of clusters is proposed in [9]. We can find the optimal clustering by examining the entire tree of clusters produced. Denote the cost of the optimal m-clustering of a tree node R as $\text{cost}(R, m)$. The optimal m-clustering of a tree node R is either the entire subtree as one cluster ($m = 1$), or the minimum over all choices of m_1-clustering over its left subtree and m_2-clustering over its right subtree ($1 < m \leq k$), where m_1, m_2 are positive integers such that $m_1 + m_2 = m$. Therefore, we can traverse the tree bottom up, recursively solving the m-clustering problem for $1 \leq m \leq k$ for each tree node. The algorithm is presented in Algorithm 14. Suppose that computing the cost of a cluster takes time t ($t = O(n^2)$ for k-median, k-means and min-sum). Since there are $O(n)$ nodes, and on each node R, computing $\text{cost}(R, 1)$ takes time t, computing $\text{cost}(R, m)(1 < m \leq k)$ takes $O(k^2)$, in total the algorithm takes time $O(nt + nk^2)$.

Notice when $\mathcal{T}$ is a multi-branch tree and not suitable for dynamic programming, we need to turn it into a 2-branch tree $\mathcal{T}'$ as follows. For each node with more than 2 children, for example, the node R with children $R_1, R_2, \ldots, R_t (t > 2)$, we first merge R_1 and R_2 into one node, then merge this node with R_3; repeat until we merge all nodes $R_1, R_2, \cdots, R_t$ into R. In this way, we get a 2-branch tree $\mathcal{T}'$ and can run dynamic programming on it. Note that each pruning in $\mathcal{T}$ has a corresponding pruning in $\mathcal{T}'$, so the minimum cost pruning of $\mathcal{T}'$ has no greater cost than the minimum cost pruning of $\mathcal{T}$.

Also note that when the cost function is center-based, such as k-median, the algorithm essentially computes a center for the node R when computing $\text{cost}(R, 1)$. So it can output the centers together with the pruning.

Algorithm 14 Dynamic Programming in Tree of Clusters

Input: A tree of clusters $\mathcal{T}$ on a data set P, distance function $d(\cdot, \cdot)$ on P, k.

1: Traverse $\mathcal{T}$ bottom up.
2: **for** each node $R \in \mathcal{T}$ **do**
3: Calculate $\text{cost}(R, 1)$. For $1 < m \leq k$, calculate $\text{cost}(R, m)$ as follows.
4: **if** R is a leaf **then**
5: $\text{cost}(R, m) = \text{cost}(R, 1)$.
6: **else**
7: $\text{cost}(R, m) = \min[\text{cost}(R_1, m_1) + \text{cost}(R_2, m_2)]$, where R_1, R_2 are R's children and $m_1 + m_2 = m$.
8: **end if**
9: **end for**
10: Traverse backwards to get the k-clustering $\mathcal{C}$ that achieves $\text{cost}(r, k)$ where r is the root.

Output: The k-clustering $\mathcal{C}$.

A.2 An Efficient Implementation of Algorithm 1

Here we show an efficient implementation of Algorithm 1, namely Algorithm 15. The Phase 1 of this implementation takes time only $O(n^3)$.

Notice at each merge step in Algorithm 1, we only need to find the two clusters with the minimum closure distance. So we hope to compute the minimum closure distance without computing all the distances between any two current clusters. First we notice the following facts.

Fact 3. *In the execution of Algorithm 1, if d is the minimum closure distance for the current clustering, then*

(1) there exist $c, p \in P$ such that $d = d(c, p)$;

(2) d is no less than the minimum closure distances in previous clusterings.

Proof. For the first claim, let c be the center of the ball in the definition of closure distance, and p be the farthest point from the center in the ball, then $d = d(c, p)$.

Algorithm 15 Efficient Implementation of Algorithm 1

Input: Data set P, distance function $d(\cdot, \cdot)$ on P.

1: Sort all the pairwise distances in ascending order.
2: **for** each $p \in P$ and $1 \leq i \leq n$ **do**
3: Compute L^p, $\chi(p, i)$. Then compute $\chi^*(p, i)$ by Equation (18).
4: **end for**
5: Let the current clustering be n singleton clusters.
6: **for** $d(p, q)$ in ascending order **do**
7: Suppose $q = L_i^p$. Check if $d(p, q)$ satisfies the three claims in Fact 4, where the third claim can be checked by verifying if $\chi^*(p, i) = -1$.
8: If so, merge all the clusters covered by $\mathbb{B}(p, d(p, q))$.
9: **end for**
10: Construct the tree $\mathcal{T}$ with points as leaves and internal nodes corresponding to the merges performed.
11: Run dynamic programming on $\mathcal{T}$ to get the minimum cost pruning $\tilde{\mathcal{C}}$.

Output: The clustering $\tilde{\mathcal{C}}$.

The second claim comes from the fact that the clusters in the current clustering are supersets of those in previous clusterings. □

Fact 3 implies that we can check in ascending order the pairwise distances no less than the minimum closure distance in the last clustering, and determine if the checked pairwise distance is the minimum closure distance in the current clustering. More specifically, suppose we have some black-box method for checking if a pairwise distance is the minimum closure distance in the current clustering, we can perform the closure linkage as follows: sort the pairwise distances in a list in ascending order; start from the first distance in the list; check if the current distance is the minimum closure distance in the current clustering; if it is, merge clusters covered by the ball defined by the checked distance; continue to check the next distance in the list. So it is sufficient to design a method to determine if a pairwise distance is the minimum closure distance in the current clustering. Our method is based on the following facts.

Fact 4. *In Algorithm 1, if $d(c, p)$ is the minimum closure distance for the current clustering, then*

(1) at least 2 clusters intersects $\mathbb{B}(c, d(c, p))$;

(2) all the clusters intersecting $\mathbb{B}(c, d(c,p))$ *are covered by* $\mathbb{B}(c, d(c,p))$*;*

(3) for any $p' \in \mathbb{B}(c, d(c,p)), q \notin \mathbb{B}(c, d(c,p)), d(c,p') < d(p',q)$.

Proof. The first claim and the third claim follow from the definition. We can prove the second claim by induction. This is trivial at the beginning. Suppose it is true up to any previous clustering, we prove it for the current clustering $\mathcal{C}'$. We need to show that for any $C' \in \mathcal{C}'$ such that $C' \cap \mathbb{B}(c, d(c,p)) \neq \varnothing$, $C' \subseteq \mathbb{B}(c, d(c,p))$. If $c \in C'$, then by definition, $C' \subseteq \mathbb{B}(c, d(c,p))$. If C' is a single point set $\{c_1\}$, then trivially $C' \subseteq \mathbb{B}(c, d(c,p))$. What is left is the case when $c \notin C'$ and C' is generated by merging clusters in a previous step. Suppose when C' is formed, the closure distance between those clusters is defined by $c_1 \in C'$ and p_1. By induction, if $c \in \mathbb{B}(c_1, d(c_1,p_1))$, c would have been merged into C' when C' is merged, which is contradictory to $c \notin C'$. So we have $c \notin \mathbb{B}(c_1, d(c_1,p_1))$, i.e. $d(c,c_1) > d(c_1,p_1)$. Then by the margin requirement of $\mathbb{B}(c_1, d(c_1,p_1))$, $d(c,q) > d(c_1,q)$ for any $q \in \mathbb{B}(c, d(c,p)) \cap C'$. This further leads to $c_1 \in \mathbb{B}(c, d(c,p))$, since otherwise by the margin requirement of $\mathbb{B}(c, d(c,p))$ and $q \in \mathbb{B}(c, d(c,p))$, we would have $d(c,q) < d(c_1,q)$. So for any point $q' \in C'$, since $d(c_1,q') \leq d(c_1,p_1) < d(c,c_1)$, we have $q' \in \mathbb{B}(c, d(c,p))$ from the margin requirement, so $C' \subseteq \mathbb{B}(c, d(c,q))$. □

Notice if a pairwise distance satisfies the three claims, then it defines a closure distance for the clusters covered. So if we check the pairwise distances in ascending order, then the first one that satisfies the three claims must be the minimum closure distance in the current clustering. So we have a method to determine if a pairwise distance is the minimum closure distance.

However, naively checking the third claim in Fact 4 takes $O(n^2)$, which is still not good enough. We can refine this step since intuitively, for every c, if $d(c,q)$ comes after $d(c,p)$ in the distance list, then when checking $d(c,q)$, we can utilize the information obtained from checking $d(c,p)$. Specifically, for every $p \in P$, define $L^p = (L^p_1, \ldots, L^p_n)$ to be a sorted list of points in P, according to their distances to p in ascending order.

Let $\chi^*(p,i)$ denote the index of the farthest point in L^p, which makes $d(p,L_i^p)$ fail the third claim in Fact 4. Formally, define $\chi^*(p,i)$ to be the maximum $j > i$ such that there exits $s \leq i$ satisfying $d(p,L_s^p) \geq d(L_s^p,L_j^p)$. If no such point L_j^p exists, let $\chi^*(p,i) = -1$. Then $d(p,L_i^p)$ satisfies the third claim if and only if $\chi^*(p,i) = -1$, thus we turn the task of checking the claim into computing $\chi^*(p,i)$. In order to use the information obtained when previously checking $d(p,L_{i-1}^p)$, we compute $\chi^*(p,i)$ from $\chi^*(p,i-1)$. By the definition of χ^*, what is new of $\chi^*(p,i)$ compared to $\chi^*(p,i-1)$ is just the maximum $j > i$ such that $d(p,L_i^p) \geq d(L_i^p,L_j^p)$. Define $\chi(p,i)$ to be the maximum $j > i$ such that $d(p,L_i^p) \geq d(L_i^p,L_j^p)$; if no such j exists, let $\chi(p,i) = -1$. Then it is easy to verify that

$$\chi^*(p,i) = \max\{\chi^*(p,i-1), \chi(p,i)\}. \tag{18}$$

It takes $O(n)$ time to compute $\chi(p,i)$, thus we can compute $\chi^*(p,i)$ for all $p \in P, 1 \leq i \leq n$ in $O(n^3)$ time. The implementation is finally summarized in Algorithm 15.

A.3 Proofs for Bounding the Number of Bad Points for k-Median

We now present the proof of Fact 1, which is used in Claims 1 and 2 for translating d' to d. For notations used, see Figure 21 in Section 2.3.1 for illustration. We begin with the following technical fact:

Fact 5. *Suppose the clustering instance is (α,ϵ)-perturbation resilient. If $\min_i |C_i| > (\frac{2}{\alpha-1}+3)\epsilon n + 1$, then $c_i' \neq c_j (\forall j \neq i)$.*

Proof. The intuition is that if $c_i' = c_j$, then under d', points in W_j should be closer to c_j' than to $c_i' = c_j$. So under d, these points are α time closer to c_j' than to c_j. This means that the distance between c_j and c_j' is no so large compared to the average distance between c_j and W_j by the triangle inequality. On the other hand, this also means that c_j has $(1-1/\alpha)d(c_j,W_j)$ more cost than c_j' on W_j. Then c_j should save this

cost on other parts of C_j. In other words, c'_j has at least $(1-1/\alpha)d(c_j, W_j)$ more cost on these points than c_j has. By the triangle inequality, the distance between c_j and c'_j is much larger than the average distance between c_j and W_j, which is contradictory.

Formally, assume for contradiction that $c'_i = c_j$. To apply the intuition described above, we first need to show $c'_j \neq c_l(\forall l)$. Clearly, $c'_j \neq c_j$, since otherwise, moving all the points in C'_j to C'_i will not increase the cost, which violates (α, ϵ)-perturbation resilience. We also know that $c'_j \neq c_l (l \neq j)$ since otherwise, there is $p \in W_j$, $d(c_l, p) = d(c'_j, p) \leq d'(c'_j, p) < d'(c'_i, p) = d(c_j, p)$, which contradicts the fact that $p \in C_j$.

Now we can apply the intuition described above to show that $c'_i = c_j$ and $c'_j \neq c_l(\forall l)$ lead to an contradiction. Note that points in $W_j \cup V_j = C_j \cap C'_j$ are closer to c'_j than to $c'_i = c_j$ under d'. Then back to d, for any $p \in W_j$, since $c'_j \neq c_l(\forall l)$, $\alpha d(c'_j, p) = d'(c'_j, p) \leq d'(c'_i, p) = d(c_j, p)$, resulting in $d(c'_j, W_j) \leq d(c_j, W_j)/\alpha$. Similarly, for any $p \in V_j$, $\alpha d(c'_j, p) = d'(c'_j, p) \leq d'(c'_i, p) = \alpha d(c_j, p)$, resulting in $d(c'_j, V_j) \leq d(c_j, V_j)$. These facts have two consequences.

First, since points in W_j are α time closer to c'_j than to c_j, the distance between c'_j and c_j is small:

$$d(c'_j, c_j) \leq \frac{d(c'_j, W_j)}{|W_j|} + \frac{d(c_j, W_j)}{|W_j|} \leq (1 + \frac{1}{\alpha}) d(c_j, W_j). \tag{19}$$

Second, since c_j is the optimal center for $C_j = W_j \cup V_j \cup M_j$, it should save a lot of cost on M_j compared to c'_j, which suggests c_j and c'_j would be far apart. Formally,

$$d(c'_j, C_j) = d(c'_j, W_j \cup V_j \cup M_j) \geq d(c_j, C_j) = d(c_j, W_j \cup V_j \cup M_j).$$

Since $d(c'_j, W_j) \leq d(c_j, W_j)/\alpha$ and $d(c'_j, V_j) \leq d(c_j, V_j)$, we have

$$\begin{aligned} d(c'_j, M_j) - d(c_j, M_j) &\geq d(c_j, W_j) - \frac{1}{\alpha} d(c_j, W_j), \\ |M_i| d(c'_j, c_j) &\geq (1 - \frac{1}{\alpha}) d(c_j, W_j). \end{aligned} \tag{20}$$

When $|C_j| > (\frac{2}{\alpha-1}+3)\epsilon n + 1$, we have $(1-1/\alpha)|W_j| > (1+1/\alpha)|M_j|$. Then Inequalities 20 and 19 lead to $d(c_j, c'_j) = 0$. This means $c_j = c'_j$ which is a contradiction to the assumptions. □

Fact 1. *Suppose the clustering instance is* (α, ϵ)*-perturbation resilient and* $\min_i |C_i| > (\frac{2}{\alpha-1}+3)\epsilon n + 1$. *If* $c_i \neq c'_i$, *then we have*

$$d'(c'_i, W_i) \geq \alpha d(c'_i, W_i \setminus \{c(c'_i)\}), \qquad d'(c_i, W_i) = d(c_i, W_i),$$
$$d'(c'_i, V_i) = \alpha d(c'_i, V_i), \qquad d'(c_i, V_i) = \alpha d(c_i, V_i),$$
$$d'(c'_i, A_i) \geq \alpha d(c'_i, A_i \setminus \{c(c'_i)\}), \qquad d'(c_i, A_i) \leq \alpha d(c_i, A_i \setminus \{c(c'_i)\}) + \alpha(1+\alpha) d(c'_i, c_i).$$

Proof. These translations can be verified by the definition of d'. In most cases, $d'(\cdot,\cdot) = \alpha d(\cdot,\cdot)$; the only exceptions are the distances between p and $c(p)$. The detailed verification is presented below.

Since $c'_i \neq c_i$, and by Fact 5, we know $c'_i \neq c_j (\forall j)$. So when translating $d'(c'_i, C)(C$ is W_i, V_i or $A_i)$, we only need to check if $c(c'_i) \in C$. For W_i, $d'(c'_i, W_i) \geq d'(c'_i, W_i \setminus \{c(c'_i)\}) = \alpha d(c'_i, W_i \setminus \{c(c'_i)\})$. For V_i, since there is no center in V_i, $d'(c'_i, V_i) = \alpha d(c'_i, V_i)$. For A_i, $d'(c'_i, A_i) \geq d'(c'_i, A_i \setminus \{c(c'_i)\}) = \alpha d(c'_i, A_i \setminus \{c(c'_i)\})$.

Now consider the sum of distances concerning c_i. For $d'(c_i, W_i)$ and $d'(c_i, V_i)$, the equations follow from the definition of d'. For A_i, if $c(c'_i) \notin A_i$, then the inequality is trivial. If $c(c'_i) \in A_i$, then

$$d'(c_i, A_i) \;=\; d'(c_i, A_i \setminus \{c(c'_i)\}) + d'(c_i, c(c'_i)) \leq \alpha d(c_i, A_i \setminus \{c(c'_i)\}) + \alpha d(c_i, c(c'_i)).$$

We have $d(c_i, c(c'_i)) \leq d(c_i, c'_i) + d(c'_i, c(c'_i))$. If c'_i is a selected bad point, then $d(c'_i, c(c'_i)) \leq \alpha d(c'_i, c_i)$. Otherwise, $c(c'_i)$ is the nearest center for c'_i, then $d(c'_i, c(c'_i)) \leq d(c'_i, c_i)$. In any case, the inequality for $d'(c_i, A_i)$ follows. □

A.4 Structural Properties of (α, ϵ)-Perturbation Resilient Min-Sum Instances

A.4.1 Proofs for Bounding the Number of Bad Points for Min-Sum

Recall the definitions of the bad points and the perturbation constructed to bound the number of bad points in Section 2.5.1.1. Suppose an intervals $[r, 2r]$ contains the costs of more than $2\epsilon n$ bad points. Let $\hat{B}$ denote a subset of $2\epsilon n$ bad points in this interval, and let $\hat{B}_i = \hat{B} \cap C_i$, $K_i = C_i \setminus \hat{B}_i$. Suppose for a bad point $p \in \hat{B}_i$, C_j is its second nearest cluster, that is, $j = \arg\min_{\ell \neq i} d(p, C_\ell)$. Denote this as $p \in D_j$. Let $\tilde{C}_i = K_i \cup D_i$. The perturbation is constructed as follows: blow up all distances by a factor of α except those within $\tilde{C}_i, 1 \leq i \leq k$. Let $\{C'_i\}$ denote the optimal clustering after perturbation. Recall the definitions of U_i, V_i, W_i and $\tilde{U}_i, \tilde{V}_i, \tilde{W}_i$, and see Figure 6 for an illustration. The following facts come from their definitions.

Fact 6. *We have $\cup_i U_i = \cup_i \tilde{U}_i$, $\cup_i V_i = \cup_i \tilde{V}_i$ and $\cup_i W_i = \cup_i \tilde{W}_i$. Furthermore,*

$$\begin{aligned}
\sum_i d(\tilde{U}_i, C_i) &\leq \beta \sum_i d(U_i, C_i), \\
\sum_i d(\tilde{V}_i, C_i) &\leq \sum_i d(V_i, C_i), \\
\sum_i d(\tilde{W}_i, C_i) &\leq \sum_i d(W_i, C_i).
\end{aligned}$$

We are ready to prove the claim needed for bounding the number of bad points.

Claim 3.(a). *The costs saved and added by moving $\{U_i, 1 \leq i \leq k\}$ satisfy*

$$\begin{aligned}
& 2\sum_i d'(U_i, C'_i \cap K_i) - 2\sum_i d'(\tilde{U}_i, \tilde{C}_i) \\
\geq\ & \frac{3}{10}\alpha \sum_i d(U_i, C_i) - \frac{2\alpha}{100}\sum_i d(W_i, C_i) - \frac{8\alpha + 16}{100} r\epsilon n.
\end{aligned}$$

Proof. Intuitively, $\sum_i d'(U_i, C'_i \cap K_i) \approx \alpha \sum_i d(U_i, C_i)$ and $\sum_i d'(\tilde{U}_i, \tilde{C}_i) \approx \sum_i d(\tilde{U}_i, C_i)$. Their difference is then roughly $(\alpha - \beta)\sum_i d(U_i, C_i)$, since $\sum_i d(\tilde{U}_i, C_i) \leq \beta \sum_i d(U_i, C_i)$.

Formally, we have

$$d'(U_i, C_i' \cap K_i) = \alpha d(U_i, C_i' \cap K_i) = \alpha d(U_i, C_i) - \alpha d(U_i, \tilde{W}_i + \hat{B}_i), \tag{21}$$

$$d'(\tilde{U}_i, \tilde{C}_i) = d(\tilde{U}_i, \tilde{C}_i) = d(\tilde{U}_i, K_i) + d(\tilde{U}_i, D_i) \leq d(\tilde{U}_i, C_i) + d(\tilde{U}_i, D_i). \tag{22}$$

Then it suffices to bound the approximation error $d(U_i, \tilde{W}_i + \hat{B}_i)$ and $d(\tilde{U}_i, D_i)$. First,

$$\begin{aligned} d(U_i, \tilde{W}_i + \hat{B}_i) &\leq \frac{|\tilde{W}_i + \hat{B}_i|}{|C_i|} d(U_i, C_i) + \frac{|U_i|}{|C_i|} d(C_i, \tilde{W}_i + \hat{B}_i) \\ &\leq \frac{3}{100} d(U_i, C_i) + \frac{1}{100} d(C_i, \tilde{W}_i + \hat{B}_i) \end{aligned}$$

where the first inequality is by Fact 7, and the second is from the fact that $|\tilde{W}_i| \leq \epsilon n, |\hat{B}_i| \leq 2\epsilon n, |U_i| \leq \epsilon n$ and $|C_i| \geq 100\epsilon n$. For the second term on the right hand side, we have $\sum_i d(C_i, \tilde{W}_i) \leq \sum_i d(W_i, C_i)$, and points in $\hat{B}_i$ has cost at most $2r$: $d(\hat{B}_i, C_i) \leq 2r|\hat{B}_i|$. So

$$\sum_i d(U_i, \tilde{W}_i + \hat{B}_i) \leq \frac{3}{100} \sum_i d(U_i, C_i) + \frac{1}{100} \sum_i d(C_i, W_i) + \frac{4r\epsilon n}{100}.$$

Similarly, for $d(\tilde{U}_i, D_i)$ we have

$$\sum_i d(\tilde{U}_i, D_i) \leq \sum_i \left[\frac{|D_i|}{|C_i|} d(\tilde{U}_i, C_i) + \frac{|\tilde{U}_i|}{|C_i|} d(C_i, D_i) \right] \leq \frac{2\beta}{100} \sum_i d(U_i, C_i) + \frac{8r\epsilon n}{100}.$$

The claim follows by summing (21) and (22) over $1 \leq i \leq k$ and plugging the last two inequalities. □

Claim 3.(b). *The costs saved and added by moving $\{V_i, 1 \leq i \leq k\}$ satisfy*

$$\begin{aligned} & 2 \sum_i d'(V_i, C_i' \cap C_i) - 2 \sum_i d'(\tilde{V}_i, \tilde{C}_i) \\ \geq\ & \frac{99}{50}(\alpha - 2) \sum_i d(V_i, C_i) - \frac{2\alpha}{100} \sum_i d(W_i, C_i) - \frac{8\alpha + 16\beta}{100} r\epsilon n. \end{aligned}$$

Proof. The intuition is similar to that of Claim 3.(a): $\sum_i d'(V_i, C_i' \cap C_i) \approx \alpha \sum_i d(V_i, C_i)$ and $\sum_i d'(\tilde{V}_i, \tilde{C}_i) \approx \sum_i d(\tilde{V}_i, C_i)$. Since $\sum_i d(V_i, C_i) \geq \sum_i d(\tilde{V}_i, C_i)$, their difference is roughly $(\alpha - 1) \sum_i d(V_i, C_i)$.

Formally, we have

$$\begin{aligned} d'(V_i, C_i' \cap C_i) &= \alpha d(V_i, C_i' \cap C_i) = \alpha d(V_i, C_i) - \alpha d(V_i, C_i \setminus C_i'), & (23) \\ d'(\tilde{V}_i, \tilde{C}_i) &= d(\tilde{V}_i, \tilde{C}_i) \leq d(\tilde{V}_i, C_i) + d(\tilde{V}_i, D_i). & (24) \end{aligned}$$

Then it suffices to bound the approximation error $d(V_i, C_i \setminus C_i')$ and $d(\tilde{V}_i, D_i)$. First,

$$\begin{aligned} d(V_i, C_i \setminus C_i') &\leq \frac{|C_i \setminus C_i'|}{|C_i|} d(V_i, C_i) + \frac{|V_i|}{|C_i|} d(C_i \setminus C_i', C_i) \\ &\leq \frac{1}{100} d(V_i, C_i) + \frac{1}{100} d(C_i \setminus C_i', C_i) \end{aligned}$$

where the first inequality is by Fact 7 and the second is from the fact that $|C_i \setminus C_i'| \leq \epsilon n, |V_i| \leq \epsilon n$ and $|C_i| \geq 100\epsilon n$. For the second term on the right hand side, we have $d(C_i \setminus C_i', C_i) \leq d(\tilde{W}_i, C_i) + d(\hat{B}_i \setminus C_i', C_i)$. Noting that $\sum_i d(\tilde{W}_i, C_i) \leq \sum_i d(W_i, C_i)$ and that the points in $\hat{B}_i$ have cost at most $2r$, we have

$$\sum_i d(V_i, C_i \setminus C_i') \leq \frac{1}{100} \sum_i d(V_i, C_i) + \frac{1}{100} \sum_i d(W_i, C_i) + \frac{4r\epsilon n}{100}.$$

Similarly, for $d(\tilde{V}_i, D_i)$ we have

$$\sum_i d(\tilde{V}_i, D_i) \leq \sum_i \left[\frac{|D_i|}{|C_i|} d(\tilde{V}_i, C_i) + \frac{|\tilde{V}_i|}{|C_i|} d(C_i, D_i) \right] \leq \frac{2}{100} \sum_i d(V_i, C_i) + \frac{8\beta r\epsilon n}{100}.$$

The claim follows by summing (23) and (24) over $1 \leq i \leq k$ and plugging the last two inequalities. □

Claim 3.(c). *The costs saved and added by moving $\{W_i, 1 \leq i \leq k\}$ satisfy*

$$\begin{aligned} & 2 \sum_i d'(W_i, C_i' \cap C_i) - 2 \sum_i d'(\tilde{W}_i, \tilde{W}_i + C_i' \cap \tilde{C}_i) \\ \geq\ & \frac{98}{50}(\alpha - 2) \sum_i d(W_i, C_i) - \frac{8\alpha + 8\beta}{100} r\epsilon n. \end{aligned}$$

Proof. The intuition is similar to that of Claim 3.(a): $\sum_i d'(W_i, C_i' \cap C_i) \approx \alpha \sum_i d(W_i, C_i)$ and $\sum_i d'(\tilde{W}_i, \tilde{W}_i + C_i' \cap \tilde{C}_i) \approx \sum_i d(\tilde{W}_i, C_i)$. Since $\sum_i d(W_i, C_i) \geq \sum_i d(\tilde{W}_i, C_i)$, their difference is roughly $(\alpha - 1) \sum_i d(W_i, C_i)$.

Formally, we have

$$d'(W_i, C'_i \cap C_i) = \alpha d(W_i, C'_i \cap C_i) = \alpha d(W_i, C_i) - \alpha d(W_i, C_i \setminus C'_i). \quad (25)$$

$$d'(\tilde{W}_i, \tilde{W}_i + C'_i \cap \tilde{C}_i) = d(\tilde{W}_i, C'_i \cap D_i + C_i \cap \tilde{C}_i) \leq d(\tilde{W}_i, C'_i \cap D_i) + d(\tilde{W}_i, C_i). \quad (26)$$

Then it suffices to bound the approximation error $d(W_i, C_i \setminus C'_i)$ and $d(\tilde{W}_i, C'_i \cap D_i)$. First,

$$\begin{aligned} d(W_i, C_i \setminus C'_i) &\leq \frac{|C_i \setminus C'_i|}{|C_i|} d(W_i, C_i) + \frac{|W_i|}{|C_i|} d(C_i \setminus C'_i, C_i) \\ &\leq \frac{1}{100} d(W_i, C_i) + \frac{1}{100} d(C_i \setminus C'_i, C_i) \end{aligned}$$

where the first inequality is by Fact 7 and the second from the fact that $|C_i \setminus C'_i| \leq \epsilon n, |W_i| \leq \epsilon n$ and $|C_i| \geq 100\epsilon n$. For the second term on the right hand side, we have $d(C_i \setminus C'_i, C_i) = d(\tilde{W}_i, C_i) + d(\hat{B}_i \setminus C'_i, C_i)$. Noting that $\sum_i d(\tilde{W}_i, C_i) \leq \sum_i d(W_i, C_i)$ and that points in $\hat{B}_i$ have cost at most $2r$, we have

$$\sum_i d(W_i, C_i \setminus C'_i) \leq \frac{2}{100} \sum_i d(W_i, C_i) + \frac{4r\epsilon n}{100}.$$

Similarly, for $d(\tilde{W}_i, C'_i \cap D_i)$ we have

$$\begin{aligned} \sum_i d(\tilde{W}_i, C'_i \cap D_i) &\leq \sum_i \left[\frac{|C'_i \cap D_i|}{|C_i|} d(\tilde{W}_i, C_i) + \frac{|\tilde{W}_i|}{|C_i|} d(C'_i \cap D_i, C_i) \right] \\ &\leq \frac{1}{100} \sum_i d(W_i, C_i) + \frac{4\beta r\epsilon n}{100}. \end{aligned}$$

The claim follows by summing (25) and (26) over $1 \leq i \leq k$ and plugging the last two inequalities. □

A.4.2 Properties of Good Points in Min-Sum

The first property is that the cost between two optimal clusters is roughly that between sufficiently large subsets of their good points (Lemma 10). To prove this, we begin with the following useful fact.

Fact 7. *For any non-empty sets A, B and C, we have $d_{\mathrm{a}}(A, B) \leq d_{\mathrm{a}}(A, C) + d_{\mathrm{a}}(C, B)$, and thus $d(A, B) \leq \frac{|B|}{|C|} d(A, C) + \frac{|A|}{|C|} d(C, B)$.*

Then we have the following useful property for good points, which shows that good points are much closer to its own cluster than to good points in any other cluster. This property then leads to Lemma 10.

Lemma 23. *For $G_A \subseteq G_i, G_B \subseteq G_j, j \neq i$, we have*

$$d_{\mathrm{a}}(G_A, C_i) \leq \gamma_{ji}\ d_{\mathrm{a}}(G_A, G_B), \text{where}\ \ \gamma_{ji} = \frac{|C_j|}{(\beta - 1/\beta)|C_i|} + \frac{1}{\beta^2 - 1}.$$

Consequently, if $\alpha > \frac{8 \max_i |C_i|}{\min_i |C_i|}$, we have $d_{\mathrm{a}}(G_A, C_i) \leq \frac{11}{50}\ d_{\mathrm{a}}(G_A, G_B)$.

Proof. For any $p \in G_A$, we have $\beta d(p, C_i) < d(p, C_j)$. It follows from Fact 7 that

$$\beta d(G_A, C_i) < d(G_A, C_j) \leq \frac{|C_j|}{|G_B|} d(G_A, G_B) + \frac{|G_A|}{|G_B|} d(C_j, G_B)$$
$$\beta d(G_B, C_j) < d(G_B, C_i) \leq \frac{|C_i|}{|G_A|} d(G_B, G_A) + \frac{|G_B|}{|G_A|} d(C_i, G_A).$$

Plug the second inequality into the first inequality, then the lemma follows. □

Lemma 10. *Suppose $\alpha > \frac{8 \max_i |C_i|}{\min_i |C_i|}$ and $W_i \subseteq G_i, W_j \subseteq G_j$. When $|C_i| \geq 50|C_i \setminus W_i|$ and $|C_j| \geq 50|C_j \setminus W_j|$, we have $d(C_i, C_j) \leq \frac{3}{2} d(W_i, W_j)$.*

Proof. By Fact 7 and Lemma 23, we have

$$d_{\mathrm{a}}(C_i, C_j) \leq d_{\mathrm{a}}(C_i, W_i) + d_{\mathrm{a}}(W_i, W_j) + d_{\mathrm{a}}(W_j, C_j) \leq (\frac{11}{50} + 1 + \frac{11}{50}) d_{\mathrm{a}}(W_i, W_j)$$

which leads to $d(C_i, C_j) \leq \frac{36}{25} \frac{|C_i||C_j|}{|W_i||W_j|} d(W_i, W_j) \leq \frac{3}{2} d(W_i, W_j)$. □

Another property of good points is that the good points from two different clusters have cost much larger than those in a third cluster have (Lemma 11). To prove this, we need to prove Lemma 24, which bounds the cost of the optimal clustering $\mathcal{C}' = \{C'_t\}$ under the perturbed distance function. Recall the definitions of the perturbation and $\mathcal{C}'$ in Section 2.5.1.2. The perturbation blows up all pairwise distances by a factor of α except the intra-cluster distances in $\tilde{\mathcal{C}}$, where $\tilde{\mathcal{C}}$ is the clustering obtained from the optimal clustering by splitting C_i into A and $C_i \setminus A$ and merging C_j and C_l. Let

$\mathcal{C}' = \{C'_i\}$ denote the optimal clustering under the perturbed distance function d', where the clusters are indexed so that C'_i corresponds to C_i and the distance between the two clustering is $\sum_i |C_i \setminus C'_i|$.

To bound the cost of $\mathcal{C}' = \{C'_t\}$, we compare it to the cost of the optimal clustering $\mathcal{C} = \{C_t\}$ before perturbation. If $\mathcal{C}' = \mathcal{C}$, then the cost is only increased by blowing up the distances between A and $C_i \setminus A$ (Claim 10). However, the optimal clustering may change after the perturbation, so we need to consider how much cost is saved by the change (Claim 11).

Intuitively, the cost saved should be small. To see this, consider a point p moved from C_s to C'_t. Then we need to pay $d'(p, C'_t)$ instead of $d(p, C_s)$. Note that p is in C_s but not C_t, so $d(p, C_s) \leq d(p, C_t)$. Also, C'_t and C_t differ only on at most ϵn points, then $d'(p, C'_t)$ is larger or comparable to $d(p, C_t)$ and thus $d(p, C_s)$.

There are two technical details in the above description. The first is to translate $d'(p, C'_t)$ to $d(p, C'_t)$. We consider two cases (as in the proof of Claim 11). If p is moved between C_j and C_l, then $d'(p, C'_t)$ is roughly $d(p, C'_t)$ since the distances between C_j, C_l are not blown up. Otherwise, $d'(p, C'_t)$ is roughly $\alpha d(p, C'_t)$. Another technical detail is to show that $d(p, C'_t)$ roughly equals $d(p, C_t)$. Since $d(p, C'_t) \geq d(p, C'_t \cap C_t)$, it suffices to show that $d(p, C'_t \cap C_t)$ is comparable to $d(p, C_t)$, where Fact 2 turns out to be useful.

Lemma 24. *Suppose* $\alpha > \frac{6 \max_i |C_i|}{\min_i |C_i|}$ *and* $\min_i |C_i| \geq 100 m_B$. *We have*

$$\sum_{t=1}^{k} d'(C'_t, C'_t) - \sum_{t=1}^{k} d(C_t, C_t) \geq 2(\alpha - 1) d(A, G_i \setminus A) - \frac{4\alpha + 8}{100} d(C_j, C_l).$$

Proof. Let $K_t = C_t \cap C'_t, A_t = C'_t \setminus C_t, M_t = C_t \setminus C'_t$. See Figure 21 for an illustration.

We have

$$\begin{aligned}
&\sum_{t=1}^{k} d'(C'_t, C'_t) - \sum_{t=1}^{k} d(C_t, C_t) \\
\geq\ & \sum_{t=1}^{k} \left[d'(K_t, K_t) + 2d'(A_t, K_t)\right] - \sum_{t=1}^{k} \left[d(K_t, K_t) + 2d(M_t, C_t)\right] \\
=\ & \left[\sum_{t=1}^{k} d'(K_t, K_t) - \sum_{t=1}^{k} d(K_t, K_t)\right] + 2\left[\sum_{t=1}^{k} d'(A_t, K_t) - \sum_{t=1}^{k} d(M_t, C_t)\right].
\end{aligned}$$

The first term on the right hand side corresponds to the cost increased by blowing up the distances within the clusters, the second term corresponds to the cost increased by moving points away. We will bound the two terms respectively in the following two claims, which then lead to the lemma.

Let $l(p)$ denote the index of the optimal cluster in $\mathcal{C}$ that p falls in: if $p \in C_t$, then $l(p) = t$. Similarly, let $l'(p)$ denote the optimal cluster in $\mathcal{C}'$ that p falls in after perturbation: if $p \in C'_t$, then $l'(p) = t$.

The first term is roughly the cost increased by blowing the distances between A_i and $C_i \setminus A_i$, which is about $2(\alpha-1)d(A, C_i \setminus A)$. However, some points in C_i may move away, so we need to exclude the cost of these points. More precisely, we only consider good points, and also exclude the cost of the good points moved away ($G_i \cap M_i$).

Claim 10.

$$\sum_{t=1}^{k} d'(K_t, K_t) - \sum_{t=1}^{k} d(K_t, K_t) \geq 2(\alpha-1)d(A, G_i \setminus A) - \frac{2(\alpha-1)}{\beta} \sum_{p \in G_i \cap M_i} d(p, C_{l'(p)}). \tag{27}$$

Proof. By the definition of the perturbation, we have

$$\begin{aligned}
&\sum_{t=1}^{k} d'(K_t, K_t) - \sum_{t=1}^{k} d(K_t, K_t) \\
\geq\ & 2d'(A \cap K_i, (G_i \setminus A) \cap K_i) - 2d(A \cap K_i, (G_i \setminus A) \cap K_i) \\
\geq\ & 2(\alpha-1)d(A \cap K_i, (G_i \setminus A) \cap K_i) \\
\geq\ & 2(\alpha-1)\left[d(A, G_i \setminus A) - d(A \cap M_i, G_i \setminus A) - d((G_i \setminus A) \cap M_i, A)\right] \\
\geq\ & 2(\alpha-1)\left[d(A, G_i \setminus A) - d(G_i \cap M_i, C_i)\right].
\end{aligned}$$

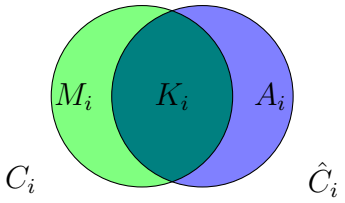


Figure 21: Notations in Lemma 24.

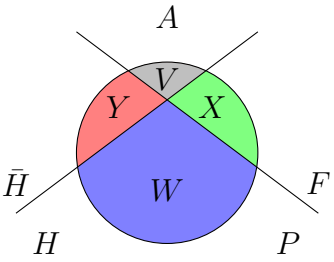


Figure 22: Notations in Claim 4.

The proof is completed by noting that for any point $p \in G_i \cap M_i$, $d(p, C_i) \leq \frac{1}{\beta} d(p, C_{l'(p)})$. □

The second term is roughly the cost increased by moving points away. Consider a point $p \in C_1$ that moves to C_2'. The new cost is $d'(p, C_2') \approx d'(p, C_2) = \alpha d(p, C_2)$, and the old cost is $d(p, C_1) \leq d(p, C_2)$, so the cost increased is roughly $(\alpha - 1)d(p, C_2)$. Note that C_2' only approximately equals C_2. Also, the above intuition does not hold for points that move between C_j and C_l since the distances between them are not blown up. These facts only decrease the bound slightly, as shown in the following claim.

Claim 11. *Let* $X = (\cup_t A_t) \setminus (A_l \cap C_j) \setminus (A_j \cap C_l)$.

$$\sum_{t=1}^{k} d'(A_t, K_t) - \sum_{t=1}^{k} d(M_t, C_t) \geq \left(\frac{98\alpha}{100} - 1 \right) \sum_{p \in X} d(p, C_{l'(p)}) - \frac{2\alpha + 4}{100} d(C_j, C_l). \quad (28)$$

Proof. We have

$$\sum_{t=1}^{k} d'(A_t, K_t) - \sum_{t=1}^{k} d(M_t, C_t) \geq \sum_{t=1}^{k} \sum_{p \in A_t} \left[d'(p, K_t) - d(p, C_{l(p)}) \right]. \quad (29)$$

Intuitively, $d'(p, K_{l'(p)})$ should be larger or comparable to $d(p, C_{l(p)})$. On one hand, $d(p, C_{l(p)}) \leq d(p, C_{l'(p)})$ since p is assigned to $C_{l(p)}$ instead of $C_{l'(p)}$ in the optimal clustering under d. On the other hand, we also know that $d(p, K_{l'(p)})$ is comparable to $d(p, C_{l'(p)})$ by Fact 2.

Before using this intuition, we first need to translate $d'(p, K_{l'(p)})$ to $d(p, K_{l'(p)})$. Since the distances between C_j and C_l is not blown up, we need to consider separately the case when p is moved between C_j and C_l. Equivalently, we divide $\cup_t A_t$ into two parts: $V = (A_j \cap C_l) \cup (A_l \cap C_j)$ and $X = (\cup_t A_t) \setminus (A_l \cap C_j) \setminus (A_j \cap C_l)$. Now we consider the two parts respectively.

Case 1: Suppose $p \in A_j \cap C_l$. By Fact 2, we have $d'(p, K_{l'(p)}) = d(p, K_j) \geq \frac{|K_j|}{|C_j|} d(p, C_j) - \frac{1}{|C_j|} d(M_j, C_j)$. Since $d(p, C_{l(p)}) = d(p, C_l) \leq d(p, C_j)$, and $d(M_j, C_j) \leq d(M_j, C_l) \leq d(C_j, C_l)$,

$$\begin{aligned} d'(p, K_{l'(p)}) - d(p, C_{l(p)}) &\geq -\frac{|M_j|}{|C_j|} d(p, C_j) - \frac{1}{|C_j|} d(C_j, C_l), \\ \sum_{p \in A_j \cap C_l} [d'(p, K_{l'(p)}) - d(p, C_{l(p)})] &\geq -\left[\frac{|M_j|}{|C_j|} + \frac{|A_j \cap C_l|}{|C_j|} \right] d(C_j, C_l). \end{aligned}$$

Since $|M_j| \leq \epsilon n$, $|A_j| \leq \epsilon n$, this is bounded by $-\frac{2}{100} d(C_j, C_l)$. A similar argument holds for $A_j \cap C_l$. So

$$\sum_{p \in V} [d'(p, K_{l'(p)}) - d(p, C_{l(p)})] \geq -\frac{4}{100} d(C_j, C_l). \tag{30}$$

Case 2: For $p \in X$, we have by Fact 2

$$d'(p, K_{l'(p)}) = \alpha d(p, K_{l'(p)}) \geq \alpha \left[\frac{|K_{l'(p)}|}{|C_{l'(p)}|} d(p, C_{l'(p)}) - \frac{1}{|C_{l'(p)}|} d(M_{l'(p)}, C_{l'(p)}) \right].$$

Then for X, since $d(p, C_{l(p)}) \leq d(p, C_{l'(p)})$ and $\frac{|K_{l'(p)}|}{|C_{l(p)}|} \geq \frac{99}{100}$, we have

$$\sum_{p \in X} [d'(p, K_{l'(p)}) - d(p, C_{l(p)})] \geq \left(\frac{99\alpha}{100} - 1 \right) \sum_{p \in X} d(p, C_{l'(p)}) - \sum_{p \in X} \frac{\alpha}{|C_{l'(p)}|} d(M_{l'(p)}, C_{l'(p)}). \tag{31}$$

Since $X \subseteq \cup_t A_t$, and $|C_t| \geq 100|A_t|$, the second term on the right hand side is

bounded by

$$\sum_t \frac{\alpha|A_t|}{|C_t|} d(M_t, C_t) \leq \frac{\alpha}{100} \sum_t d(M_t, C_t) = \frac{\alpha}{100} \sum_{p \in \cup_t A_t} d(p, C_{l(p)})$$
$$= \frac{\alpha}{100} \left[\sum_{p \in V} d(p, C_{l(p)}) + \sum_{p \in V} d(p, C_{l(p)}) \right]$$
$$\leq \frac{\alpha}{100} \left[\sum_{p \in V} d(p, C_{l'(p)}) + 2d(C_i, C_j) \right]. \tag{32}$$

The claim follows from the inequalities (30), (31), and (32). □

The proof is completed by combining the two claims. □

Lemma 11. *Suppose* $\alpha > \frac{8 \max_i |C_i|}{\min_i |C_i|}$ *and* $\epsilon < \frac{\min_i |C_i|}{600n}$. *For any three different optimal clusters* C_i, C_j, *and* C_l, *and any* $A \subset G_i$, $\frac{18}{5} d(A, G_i \setminus A) < d(G_j, G_l)$. *Consequently,* $\frac{9}{5} d(G_i, G_i) < d(G_j, G_l)$.

Proof. The key idea is as follows. Let $\tilde{\mathcal{C}}$ denote the clustering obtained from the optimal clustering by splitting C_i into A and $C_i \setminus A$ and merging C_j and C_l, i.e. $\tilde{\mathcal{C}} = \{A, C_i \setminus A, C_j \cup C_l\} \cup \{C_t, t \neq i, j, l\}$. Suppose we construct a perturbation that favors the clustering $\tilde{\mathcal{C}}$: blow up all pairwise distances by a factor of α except the intra-cluster distances in $\tilde{\mathcal{C}}$. Let $\mathcal{C}' = \{C'_i\}$ denote the optimal clustering under the perturbed distance function d', where the clusters are indexed so that C'_i corresponds to C_i and the distance between the two clustering is $\sum_i |C_i \setminus C'_i|$. By (α, ϵ)-perturbation resilience, we know that $\mathcal{C}'$ is different from $\tilde{\mathcal{C}}$ and has no greater cost than $\tilde{\mathcal{C}}$. We then show that compared to the optimal cost under the original distances, the cost of $\tilde{\mathcal{C}}$ under perturbed distances d' is larger by at most $O(d(C_j, C_l)) = O(d(G_j, G_l))$, while the cost of $\mathcal{C}'$ under perturbed distances d' is larger by roughly $O(\alpha) d(A, G_i \setminus A)$. These then leads to the first statement.

More precisely, the cost of $\tilde{\mathcal{C}}$ under d' is larger than that of $\mathcal{C}$ under d by at most $2d(C_j, C_l)$. For $\mathcal{C}'$, we have $\sum_{t=1}^k d'(C'_t, C'_t) - \sum_{t=1}^k d(C_t, C_t) \geq 2(\alpha - 1) d(A, G_i \setminus A) -$

$\frac{4\alpha+8}{100}d(C_j, C_l)$ by Lemma 24. Since $\mathcal{C}'$ has smaller cost than $\tilde{\mathcal{C}}$, we have

$$2(\alpha-1)d(A, G_i \setminus A) - \frac{4\alpha+8}{100}d(C_j, C_l) \leq 2d(C_j, C_l).$$

When $\alpha > \frac{8\max_i |C_i|}{\min_i |C_i|}$, we have $\frac{27}{5}d(A, G_i \setminus A) \leq d(C_j, C_l)$. By Lemma 10, $d(C_j, C_l) \leq \frac{3}{2}d(G_j, G_l)$, which then leads to the first part of the lemma.

The second part of the lemma follows from the fact that $\sum_{A \subseteq G_i} d(A, G_i \setminus A) = \frac{2^{|G_i|}}{2}d(G_i, G_i)$. □

A.4.3 Properties of Potential Good Points in Min-Sum

A key property of the potential good points is that the cost between any point p and the potential good points in a sufficiently large set A is roughly the cost between p and any sufficiently large subset H of A (Lemma 12). In its proof in Section 2.5.1.2, we need the following claim.

Claim 4. *Suppose $H \subseteq A$ such that $|A \setminus H| \leq m_B$. Let $F = F(A), P = P(A), \bar{H} = A \setminus H$. Let $W = H \cap P, V = \bar{H} \cap F, X = F \cap H, Y = \bar{H} \cap P$. See Figure 22 for illustration. If $|A| \geq 20m_B$, then $d(W, X) \geq d(Y, W+Y)$.*

Proof. The claim is true if $Y = \emptyset$. Otherwise, by definition of the potential bad points $F = F(A)$, we have $d_{\mathrm{a}}(X, A) \geq d_{\mathrm{a}}(Y, A)$. By definition,

$$|A|d_{\mathrm{a}}(X, A) = |W|d_{\mathrm{a}}(X, W) + |V|d_{\mathrm{a}}(X, V) + |Y|d_{\mathrm{a}}(X, Y) + |X|d_{\mathrm{a}}(X, X), \quad (33)$$

$$|A|d_{\mathrm{a}}(Y, A) = |W+Y|d_{\mathrm{a}}(Y, W+Y) + |V|d_{\mathrm{a}}(Y, V) + |X|d_{\mathrm{a}}(Y, X). \quad (34)$$

To compare $d(X, W)$ and $d(Y, W+Y)$, we need to bound the other terms in (33) and (34). By Fact 7,

$$d_{\mathrm{a}}(X, V) \leq d_{\mathrm{a}}(X, W) + d_{\mathrm{a}}(W, Y) + d_{\mathrm{a}}(Y, V),$$

$$d_{\mathrm{a}}(X, Y) \leq d_{\mathrm{a}}(X, W) + d_{\mathrm{a}}(Y, W), \quad d_{\mathrm{a}}(X, X) \leq 2d_{\mathrm{a}}(X, W).$$

Now we plug these into (33), and then plug (33) and (34) into $d_{\mathrm{a}}(X, A) \geq d_{\mathrm{a}}(Y, A)$. Since $d(W, Y) \leq d(Y, W + Y)$ and $d_{\mathrm{a}}(Y, X) \geq 0$, we have

$$\big[|W| - |Y| - |V|\big] d(Y, W + Y) \leq \big[|W| + 2|X| + |Y| + |V|\big] \frac{|Y|}{|X|} d(X, W).$$

Since $|X + V| = |F| = 2m_B$ and $|Y + V| = |A \setminus H| \leq m_B$, we have $\frac{|Y|}{|X|} \leq 1/2$. Then the lemma follows from the fact that $|A| \geq 20m_B, |F| = 2m_B$ and $|A \setminus H| \leq m_B$. □

A.5 Approximation Algorithm for (α, ϵ)-Perturbation Resilient Min-Sum

A.5.1 Proofs for Tree Construction for Min-Sum

To show that the robust average linkage algorithm keeps the laminarity of the clustering $\mathcal{L}$ with respect to the optimal clustering (after removing the bad points), we need to show the following claim.

Claim 6. *(a) Suppose $A \in \mathcal{L}, A \cap G \subsetneq G_i$, and $D \in \mathcal{L}, D \cap G \subsetneq G_j (j \neq i)$. Then there exists $A' \neq A$ in $\mathcal{L}$ such that $A' \cap G \subsetneq G_i$ and $d_{ra}(A, A') < d_{ra}(A, D)$.*
(b) Suppose $A \in \mathcal{L}, A \cap G \subsetneq G_i$, and $D \in \mathcal{L}$, $D \cap G$ is the union of good points in several optimal clusters. Then there exists $A' \neq A$ in $\mathcal{L}$ such that $A' \cap G \subsetneq G_i$ and $d_{ra}(A, A') < d_{ra}(A, D)$.

Proof. (a) The claim follows from the following three statements:

1. $d_{\mathrm{a}}(A \cap G, A' \cap G) < \frac{1}{2} d_{\mathrm{a}}(A \cap G, D \cap G)$;
2. $d_{ra}(A, A') \leq \frac{7}{5} d_{\mathrm{a}}(A \cap G, A' \cap G)$;
3. $\frac{9}{10} d_{\mathrm{a}}(A \cap G, D \cap G) \leq d_{ra}(A, D)$.

1. For simplicity, let $G_A = A \cap G, G_D = D \cap G$. From Lemma 23, we have

$$d_{\mathrm{a}}(G_A, C_i) \leq \gamma_{ji}\ d_{\mathrm{a}}(G_A, G_D), \text{where } \ \gamma_{ji} = \frac{|C_j|}{(\beta - 1/\beta)|C_i|} + \frac{1}{\beta^2 - 1}.$$

Since $d(G_A, G_i \setminus G_A) \le d(G_A, C_i)$, we have

$$d_{\mathrm{a}}(G_A, G_i \setminus G_A) \le \frac{|C_i|}{|G_i \setminus G_A|} d_{\mathrm{a}}(G_A, C_i) \le \gamma_{ji} \frac{|C_i|}{|G_i \setminus G_A|} d_{\mathrm{a}}(G_A, G_D) \le \frac{1}{2} d_{\mathrm{a}}(G_A, G_D)$$

where the last step follows from $\alpha \ge 6\frac{\max_i |C_i|}{\min_i |C_i|} + 2$, $|G_i \setminus A|$ is at least $\frac{1}{2}\min_i |C_i| - m_B$.

2. By Lemma 12 and the fact that $|A| \ge \frac{1}{2}\min_i |C_i|, |A'| \ge \frac{1}{2}\min_i |C_i|$ and $\min_i |C_i| > 100 m_B$, we have

$$d(P(A), P(A')) \le \frac{10}{9} d(P(A), A' \cap G) \le \frac{100}{81} d(A \cap G, A' \cap G).$$

Then the claim follows from the fact that $|P(A)| \ge \frac{48}{50}|A|, |P(A')| \ge \frac{48}{50}|A'|$.

3. For simplicity, let $G_A = A \cap G, G_D = D \cap G$. Divide G_A into two parts: $W_A = G_A \cap P(A)$ and $X_A = G_A \cap F(A)$. Define W_D and X_D similarly. See Figure 23 for illustration.

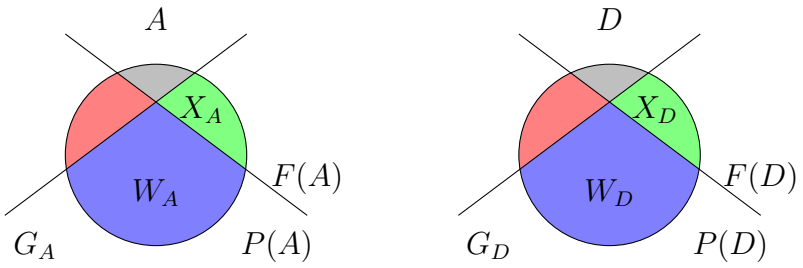


Figure 23: Notations in Claim 6.

To show $d(G_A, G_D) \le O(1) d(P(A), P(D))$, it suffices to show $d(G_A, G_D) \le O(1) d(W_A, W_D)$. Since $d(G_A, G_D) \le d(W_A, W_D) + d(G_A, X_D) + d(G_D, X_A)$, we only need to bound $d(G_A, X_D)$ and $d(G_D, X_A)$. By Fact 7 and Lemma 23 we have

$$d_{\mathrm{a}}(G_A, X_D) \le d_{\mathrm{a}}(G_A, G_D) + d_{\mathrm{a}}(G_D, X_D) \le (1 + \frac{11}{50}) d_{\mathrm{a}}(G_A, G_D).$$

Since $|X_D| \le 2m_B$ and $|D| \ge \frac{1}{2}\min_i |C_i| \ge 50 m_B$, we have $d(G_A, X_D) \le \frac{61}{50}\frac{|X_D|}{|G_D|} d(G_A, G_D) \le \frac{1}{20} d(G_A, G_D)$. Similarly, $d(G_D, X_A) \le \frac{1}{20} d(G_A, G_D)$. Therefore,

$$d(G_A, G_D) \le d(W_A, W_D) + + d(G_A, X_D) + d(G_D, X_A) \le d(W_A, W_D) + \frac{1}{10} d(G_A, G_D)$$

which leads to $\frac{9}{10}d(G_A, G_D) \leq d(W_A, W_D) \leq d(P(A), P(D))$. Then the claim follows from the fact that $|G_A| \geq |P(A)|, |G_D| \geq |P(D)|$.

(b) The proof idea is similar to that for (a). The only difference is the proof for $d_{\mathrm{a}}(A \cap G, A' \cap G) < \frac{1}{2}d_{\mathrm{a}}(A \cap G, D \cap G)$. Since $D \cap G = \cup_{j \in I_D} G_j$, it suffices to show that $d_{\mathrm{a}}(A \cap G, A' \cap G) < \frac{1}{2}d_{\mathrm{a}}(A \cap G, G_j)$ for any $j \in I_D$, which can be proved by the same argument in Claim 6. □

A.5.2 Proofs for Getting the Pruning Close to the Optimal Clustering

The same argument as that for Claim 8 leads to a corollary for the general case when multiple clusters are merged.

Corollary 1. *Let $I \subseteq [k]$. Suppose for any $t \in I$, $|C_t| \geq 100m_B$, and C'_t contains all good points in C_t but no good points in other optimal clusters. Then*

$$d_{\mathrm{rs}}\left(\cup_{t \in I} C'_t\right) - \sum_{t \in I} d(G_t, G_t) \geq \left(\frac{4}{3} - \frac{4}{\beta}\right) \sum_{s \neq t \in I} d(G_t, G_s).$$

Lemma 14. *Suppose the pruning $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ in the tree $\mathcal{T}$ assigns all good points correctly. Then $\mathcal{C}'$ is the minimum robust min-sum cost pruning in the tree.*

Proof. First, by Lemma 11, good points from different clusters are far apart while good points in the same cluster are close. Second, by Claim 7 and Corollary 1, the cost of good points can be approximated by the cost of the potential good points (the robust min-sum cost). We now use the above lemmas to show that $\mathcal{C}'$ has minimum robust min-sum cost, so that we can use dynamic programming on the tree to get the pruning.

Suppose a pruning $\mathcal{P}$ is obtained by splitting h clusters in $\mathcal{C}'$ and at the same time joining some other clusters into g unions. Specifically, for $1 \leq i \leq h$, split C'_i into $m_i \geq 2$ clusters $P_{i,1}, \ldots, P_{i,m_i}$; after that, merge $C'_{h+1}, \ldots, C'_{h+l_g}$ into g unions, i.e. for $1 \leq j \leq g$, $l_0 = 0$, merge $l_j - l_{j-1} \geq 2$ clusters $C'_{h+l_{j-1}+1}, \ldots, C'_{h+l_j}$ into a union U_j;

the other clusters in $\mathcal{C}'$ remain the same in $\mathcal{P}$. Since the number of clusters is still k, we have $\sum_i m_i - h = l_g - g$.

By Claim 7, the cost saved by splitting the h clusters is

$$\sum_{1\le i\le h} d_{\mathrm{rs}}(C'_i) - \sum_{1\le i\le h}\sum_{1\le p\le m_i} d_{\mathrm{rs}}(P_{i,p}) \le \sum_{1\le i\le h} d_{\mathrm{rs}}(C'_i) \le \sum_{1\le i\le h} d(G_i, G_i). \tag{35}$$

The cost increased by joining clusters is

$$\begin{aligned}
&\sum_{1\le j\le g}\left[d_{\mathrm{rs}}(U_j) - \sum_{h+l_{j-1}<t\le h+l_j} d_{\mathrm{rs}}(C'_t)\right] \\
&\ge \sum_{1\le j\le g}\left[d_{\mathrm{rs}}(U_j) - \sum_{h+l_{j-1}<t\le h+l_j} d(G_t, G_t)\right] \\
&\ge \sum_{1\le j\le g}\left[\sum_{h+l_{j-1}<t\ne s\le h+l_j} (\frac{4}{3} - \frac{4}{\beta}) d(G_t, G_s)\right]
\end{aligned} \tag{36}$$

where the first inequality follows from Claim 7, and the second inequality follows from Corollary 1. To prove $\mathcal{C}'$ is the minimum cost pruning, we need to show that the saved cost (35) is less than the increased cost (36). Since by Lemma 11, each term in (36) is larger than any term in (35), it is sufficient to show that the number of the terms in (36) is no less than the number of the terms in (35), that is $\sum_{1\le j\le g} \binom{l_j - l_{j-1}}{2} \ge h$. We have $\sum_j \binom{l_j - l_{j-1}}{2} = \frac{1}{2}\sum_j (l_j - l_{j-1})(l_j - l_{j-1} - 1) \ge \sum_j (l_j - l_{j-1} - 1) = l_g - g$, where the inequality is from $l_j - l_{j-1} \ge 2$. Since $m_i \ge 2$, $l_g - g = \sum_{i-1}^{h} m_i - h \ge h$, which completes the proof. □

A.5.3 Getting A Constant Factor Approximation

To get a good approximation from a clustering $\mathcal{C}'$ that assigns all good points correctly, Algorithm 7 reassigns each point p to the index i that minimizes the cost between p and the potential good points in C'_i. The following lemma shows that after this reassignment, all good points are still assigned correctly.

Lemma 15. *For any $p \in G_i$, any $j \ne i$, $d(p, P(C'_j)) > d(p, P(C'_i))$.*

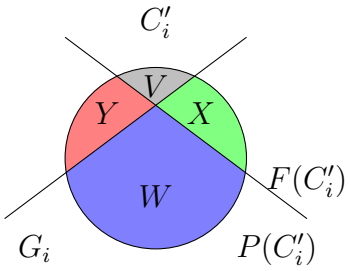


Figure 24: Notations in Lemma 15 and Claim 9.

Proof. Let $W_i = G_i \cap P(C_i')$ denote the good points that are also potential good points, and let $Z_i = C_i \setminus W_i$ denote all other points in C_i. See Figure 24 for an illustration. By Lemma 12, $d(p, P(C_i')) \approx d(p, C_i)$. By definition of good points, $\beta d(p, C_i) \leq d(p, C_j)$. So it suffices to show that $d(p, P(C_j'))$ is not so small compared to $d(p, C_j)$. Since $W_j \subseteq P(C_j')$, it suffices to prove that $d(p, W_j)$ is large compared to $d(p, Z_j)$.

First, by triangle inequality, $d(p, Z_j) \leq \frac{|Z_j|}{|W_j|} d(p, W_j) + \frac{1}{|W_j|} d(Z_j, W_j)$. Also, $d(Z_j, W_j) \leq d(C_j, W_j) \leq \frac{1}{\beta} d(C_i, W_j)$ by definition of good points. Furthermore, $d(C_i, W_j) \leq |W_j| d(p, C_i) + |C_i| d(p, W_j)$. So

$$\begin{aligned} d(p, Z_j) &\leq \left(\frac{|Z_j|}{|W_j|} + \frac{|C_i|}{\beta |W_j|} \right) d(p, W_j) + \frac{1}{\beta} d(p, C_i) \\ &\leq \left(\frac{|Z_j|}{|W_j|} + \frac{|C_i|}{\beta |W_j|} \right) d(p, W_j) + \frac{1}{\beta^2} [d(p, Z_j) + d(p, W_j)]. \end{aligned}$$

Therefore, we have $d(p, Z_j) \leq \frac{1}{3} d(p, W_j)$, since $|Z_j| \leq 4m_B, |W_j| \geq \frac{95}{100} |C_j| \geq 95 m_B$. This leads to $d(p, W_j) \geq \frac{3}{4} d(p, C_j)$. Then the lemma follows from

$$d(p, P(C_j')) \geq d(p, W_j) \geq \frac{3}{4} d(p, C_j) \geq \frac{3}{4} \beta d(p, C_i) \geq \frac{3}{4} \beta d(p, G_i) \geq \frac{30}{44} \beta d(p, P(C_i'))$$

where the last step follows from Lemma 12. □

Recall that A_i are all the bad points reassigned to index i by Algorithm 7. To bound the cost of the clustering after reassignment, we need to bound $d(A_i, G_i)$ as follows.

Claim 9. $\sum_i d(A_i, G_i) \leq \frac{r^2}{(r-5)^2} \sum_i d(C_i, C_i) - \frac{r^2-1}{(r-5)^2} \sum_i d(G_i, G_i)$, *where* $r = \frac{\min_i |C_i|}{m_B}$.

Proof. Let $W_i = P(C_i') \cap G_i$. See Figure 24 for an illustration. By Fact 6, we have

$$d(A_i, G_i) \leq \frac{|G_i|}{|W_i|} d(A_i, W_i) + \frac{|A_i|}{|W_i|} d(G_i, W_i) \leq \frac{|G_i|}{|W_i|} d(A_i, P(C_i')) + \frac{|A_i|}{|W_i|} d(G_i, G_i). \quad (37)$$

So it suffices to bound $d(A_i, P(C_i'))$. Fix $p \in A_i$, and suppose $p \in C_j$. We have

$$\begin{aligned} d(p, P(C_i')) &\leq d(p, P(C_j')) \leq \frac{|W_j| + |Y_j|}{|W_j| - |X_j|} d(p, G_j) \\ &\leq \frac{|C_j|}{|C_j| - 2|X_j| - |Y_j|} d(p, G_j) = \frac{r}{r-5} d(p, G_j) \end{aligned}$$

where the second step follows from Lemma 12 and the last from $|X_j| \leq 2m_B$ and $|Y_j| \leq m_B$. Then

$$\begin{aligned} \sum_{i=1}^{k} d(A_i, P(C_i')) &\leq \frac{r}{r-5} \sum_j \sum_{p \in (\cup_i A_i) \cap C_j} d(p, G_j) = \frac{r}{r-5} \sum_{j=1}^{k} d(B_j, G_j) \\ &\leq \frac{r}{r-5} \sum_{j=1}^{k} [d(C_j, C_j) - d(G_j, G_j)]. \end{aligned} \quad (38)$$

The claim follows from the inequalities (37) and (38) and $|X_i| \leq 2m_B, |A_i| \leq m_B$. □

APPENDIX B

DISTRIBUTED CLUSTERING

B.1 Proofs for Section 3.2.1

The proof of Lemma 16 follows from the analysis in [45], although not explicitly stated there. We begin with the following theorem for uniform sampling on a function space. The theorem is from [45] but rephrased for convenience.

Theorem 18 (Theorem 6.9 in [45]). *Let F be a set of functions from P to $\mathbf{R}_{\geq 0}$, and let $\epsilon \in (0,1)$. Let S be a sample of*

$$|S| = \frac{c}{\epsilon^2}(\dim(F,P) + \log\frac{1}{\delta})$$

i.i.d items from P, where c is a sufficiently large constant. Then, with probability at least $1-\delta$, for any $f \in F$ and any $r \geq 0$,

$$\left| \frac{\sum_{p\in P, f(p)\leq r} f(p)}{|P|} - \frac{\sum_{q\in S, f(q)\leq r} f(q)}{|S|} \right| \leq \epsilon r.$$

Proof of Lemma 16. Without loss of generality, assume $m_p \in \mathbf{N}^+$. Define G as follows: for each $p \in P$, include m_p copies $\{p_i\}_{i=1}^{m_p}$ of p in G and define $f(p_i) = f(p)/m_p$. Then S is equivalent to a sample draw i.i.d. and uniformly at random from G. We now apply Theorem 18 on G and $r = \max_{f\in F, p'\in G} f(p')$. By Theorem 18, we know that for any $f \in F$,

$$\left| \frac{\sum_{p'\in G} f(p')}{|G|} - \frac{\sum_{q'\in S} f(q')}{|S|} \right| \leq \epsilon \max_{p'\in G} f(p'). \tag{39}$$

The lemma then follows from multiplying both sides of (39) by $|G| = \sum_{p\in P} m_p$. Also note that the dimension $\dim(F,G)$ is the same as that of $\dim(F,P)$ as pointed out by [45]. □

Lemma 25. *If* $d(p,b_p)^2/\epsilon \leq |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$, *then*

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq 8\epsilon \min\{d(p,\mathbf{x})^2, d(b_p,\mathbf{x})^2\}.$$

Proof. We first have by triangle inequality

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq d(p,b_p)[d(p,\mathbf{x}) + d(b_p,\mathbf{x})].$$

Then by $d(p,b_p)^2/\epsilon \leq |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$,

$$d(p,b_p) \leq \epsilon[d(p,\mathbf{x}) + d(b_p,\mathbf{x})].$$

Therefore, we have

$$\begin{aligned}
|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| &\leq d(p,b_p)[d(p,\mathbf{x}) + d(b_p,\mathbf{x})] \leq \epsilon[d(p,\mathbf{x}) + d(b_p,\mathbf{x})]^2 \\
&\leq 2\epsilon[d(p,\mathbf{x})^2 + d(b_p,\mathbf{x})^2] \leq 2\epsilon[d(p,\mathbf{x})^2 + (d(p,\mathbf{x}) + d(p,b_p))^2] \\
&\leq 2\epsilon[d(p,\mathbf{x})^2 + 2d(p,\mathbf{x})^2 + 2d(p,b_p)^2] \leq 6\epsilon d(p,\mathbf{x})^2 + 4\epsilon d(p,b_p)^2 \\
&\leq 6\epsilon d(p,\mathbf{x})^2 + 4\epsilon^2 |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|
\end{aligned}$$

for sufficiently small ϵ. Then

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq \frac{6\epsilon}{1-4\epsilon^2} d(p,\mathbf{x})^2 \leq 8\epsilon d(p,\mathbf{x})^2.$$

Similarly, $|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq 8\epsilon d(b_p,\mathbf{x})^2$. The lemma follows from the last two inequalities. □

Lemma 26 (Corollary 15.4 in [45])**.** *Let* $0 < \delta < 1/2$, *and* $t \geq c|B| \log \frac{|B|}{\delta}$ *for a sufficiently large c. Then with probability at least* $1-\delta$, $\forall b \in B_i, \sum_{q \in P_b \cap S} w_q \leq 2|P_b|$.

B.2 Complete Experimental Results

Here we present the results of all the data sets over different network topologies and data partition methods.

Figure 25 shows the results of all the data sets on random graphs. The first column of Figure 25 shows that our algorithm and COMBINE perform nearly the same in

the uniform data partition. This is not surprising since our algorithm reduces to the COMBINE algorithm when each local site has the same cost and the two algorithms use the same amount of communication. In this case, since in our algorithm the sizes of the local samples are proportional to the costs of the local solutions, it samples the same number of points from each local data set. This is equivalent to the COMBINE algorithm with the same amount of communication. In the similarity-based partition, similar results are observed as this partition method also leads to balanced local costs. However, in the weighted partition where local sites have significantly different contributions to the total cost, our algorithm outperforms COMBINE. It improves the k-means cost by $2\% - 5\%$, and thus saves $10\% - 30\%$ communication cost to achieve the same approximation ratio.

Figure 26 shows the results of all the data sets on grid and preferential graphs. Similar to the results on random graphs, our algorithm performs nearly the same as COMBINE in the similarity-based partition and outperforms COMBINE in the weighted partition and degree-based partition. Furthermore, Figure 25 and 26 also show that the performance of our algorithm merely changes over different network topologies and partition methods.

Figure 27 shows the results of all the data sets on the spanning trees of the random graphs and Figure 28 shows those on the spanning trees of the grid and preferential graphs. Compared to the algorithm of Zhang et al., our algorithm consistently shows much better performance on all the data sets in different settings. It improves the k-means cost by $10\% - 30\%$, and thus can achieve even better approximation ratio with only 10% communication cost. This is because the algorithm of Zhang et al. constructs coresets from component coresets and needs larger coresets to prevent the accumulation of errors. Figure 27 also shows that although their costs decrease with the increase of the communication, the decrease is slower on larger graphs (e.g., as in the experiments for YearPredictionMSD). This is due to the fact that the spanning

tree of a larger graph has larger height, leading to more accumulation of errors. In this case, more communication is needed to prevent the accumulation.

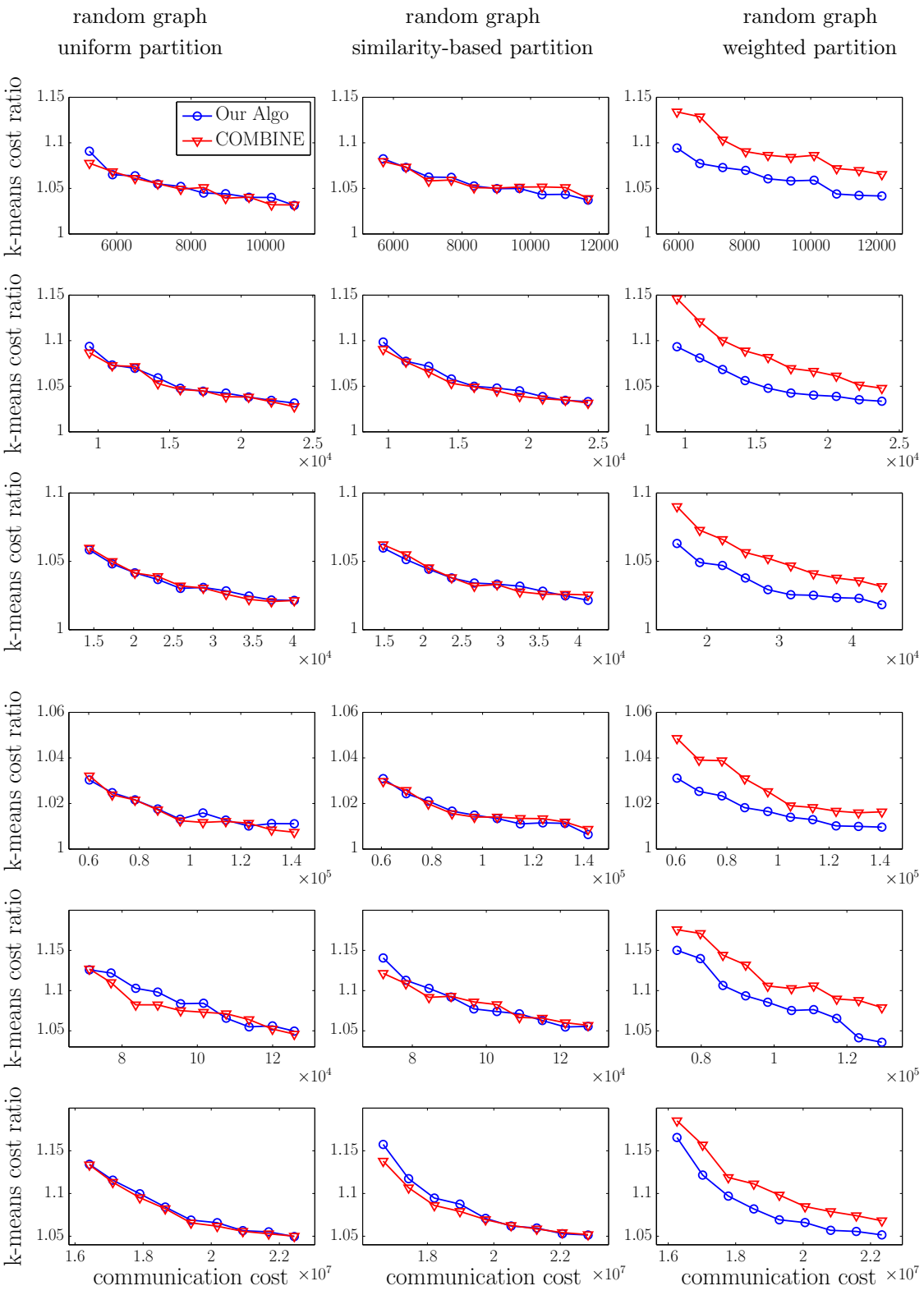


Figure 25: k-means cost on random graphs. Columns: random graph with uniform partition, random graph with similarity-based partition, and random graph with weighted partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

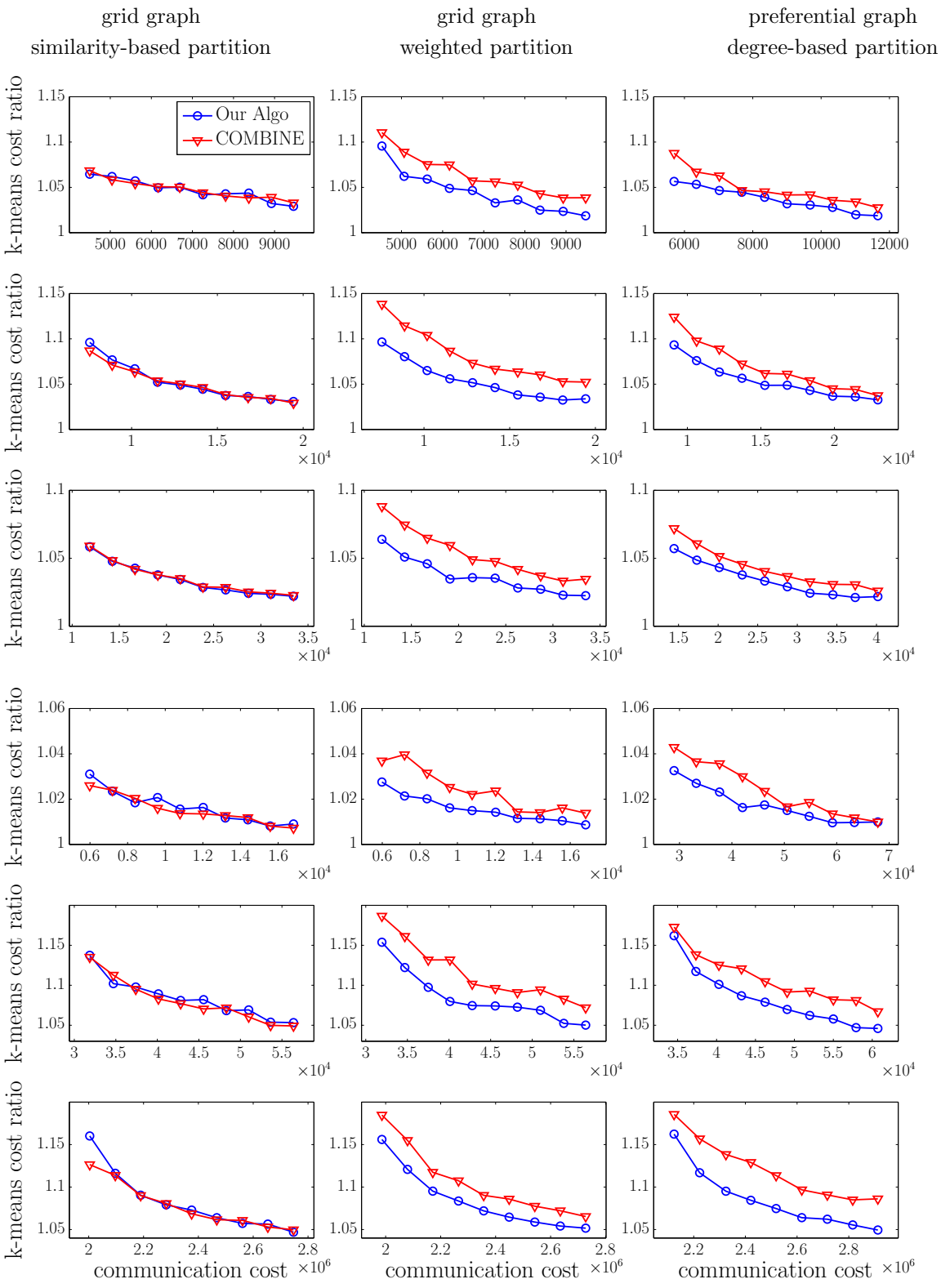


Figure 26: k-means cost on grid and preferential graphs. Columns: grid graph with similarity-based partition, grid graph with weighted partition, and preferential graph with degree-based partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

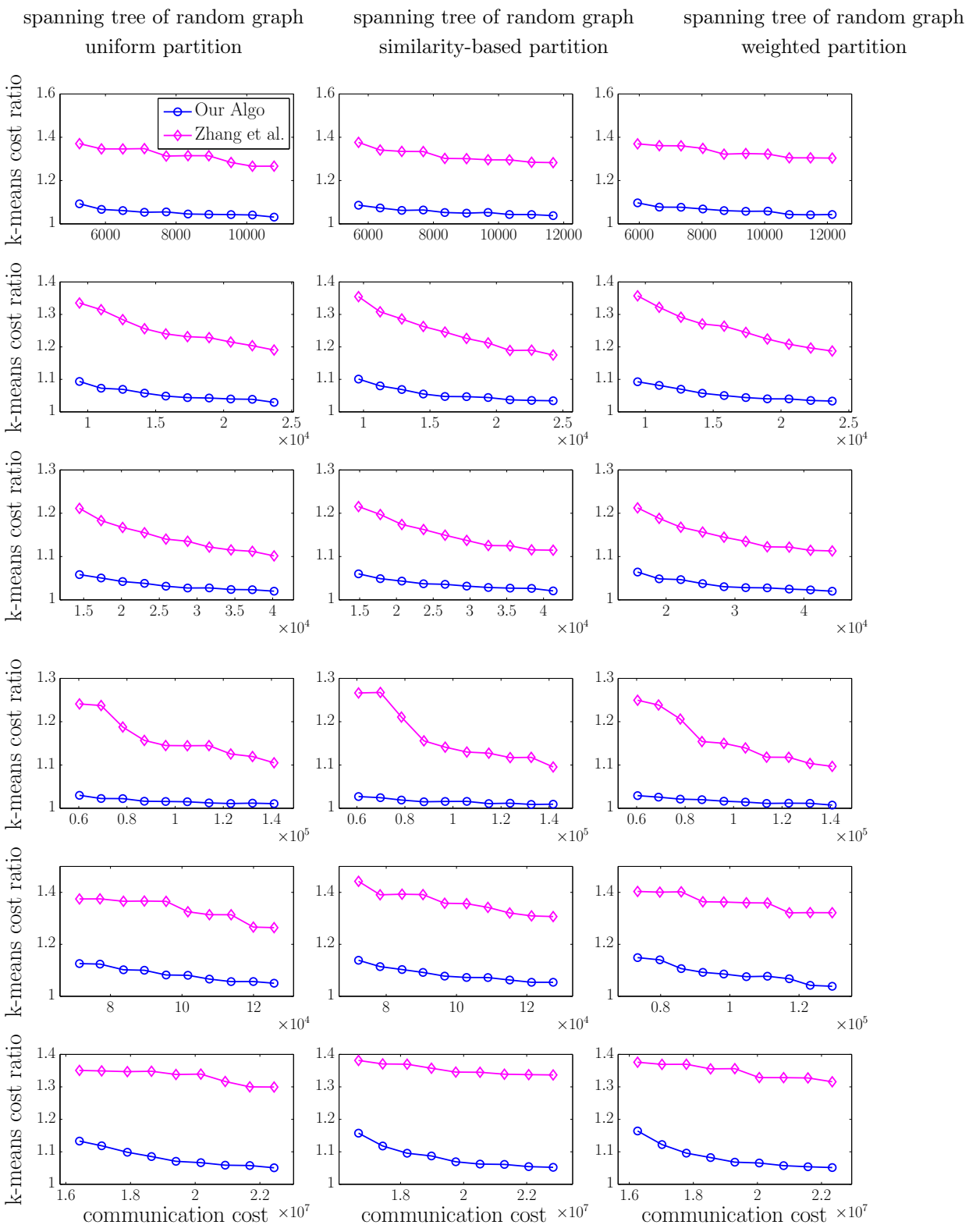


Figure 27: k-means cost on the spanning trees of the random graphs. Columns: random graph with uniform partition, random graph with similarity-based partition, and random graph with weighted partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

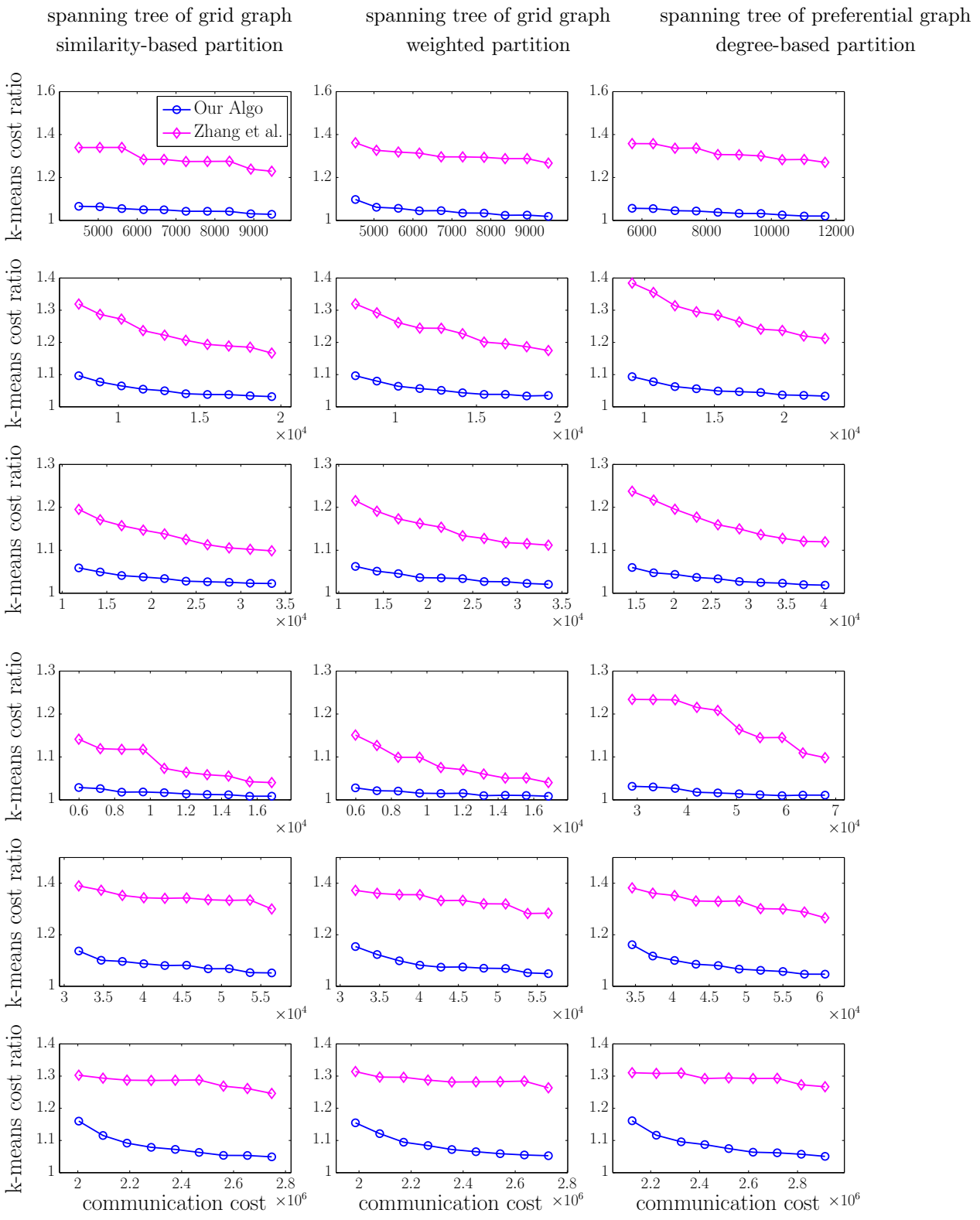


Figure 28: k-means cost on the spanning trees of the grid and preferential graphs. Columns: grid graph with similarity-based partition, grid graph with weighted partition, and preferential graph with degree-based partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

B.3 Distributed k-Means Clustering of High Dimensional Data

B.3.1 Proof of Lemma 18

Lemma 18. *Let $A \in \mathbf{R}^{n\times d}$ be an $n \times d$ matrix with singular value decomposition $A = UDV^\top$. Let $\epsilon \in (0,1]$ and $r,t \in \mathbf{N}_+$ with $d-1 \geq t \geq r + \lceil r/\epsilon \rceil - 1$, and let $\tilde{A} = AV^{(t)}(V^{(t)})^\top$. Then for any matrix X with d rows and $\|X\|_F^2 \leq r$, we have*

$$\|(A-\tilde{A})X\|_F^2 = \|AX\|_F^2 - \|\tilde{A}X\|_F^2 \leq \epsilon \sum_{i=r+1}^{d} \sigma_i^2(A).$$

Proof. The proof follows the idea in the proof of Lemma 6.1 in [46].

For convenience, let $\overline{D^{(t)}}$ denote the diagonal matrix that contains the first t diagonal entries in D and is 0 otherwise. Then $\tilde{A} = U\overline{D^{(t)}}V^\top$ We first have

$$\begin{aligned}
\|AX\|_F^2 - \|\tilde{A}X\|_F^2 &= \|UDV^\top X\|_F^2 - \|U\overline{D^{(t)}}V^\top X\|_F^2 \\
&= \|DV^\top X\|_F^2 - \|\overline{D^{(t)}}V^\top X\|_F^2 \\
&= \|(D-\overline{D^{(t)}})V^\top X\|_F^2 \\
&= \|U(D-\overline{D^{(t)}})V^\top X\|_F^2 \\
&= \|AX - \tilde{A}X\|_F^2.
\end{aligned}$$

where the second and fourth equalities follow since U has orthonormal columns, and the third equality follows since for $M = V^\top X$ we have

$$\begin{aligned}
\|DM\|_F^2 - \|\overline{D^{(t)}}M\|_F^2 &= \sum_{i=1}^{d}\sum_{j=1}^{d} \sigma_i^2(A)m_{ij}^2 - \sum_{i=1}^{t}\sum_{j=1}^{d} \sigma_i^2(A)m_{ij}^2 \\
&= \sum_{i=t+1}^{d}\sum_{j=1}^{d} \sigma_i^2(A)m_{ij}^2 = \|(D-\overline{D^{(t)}})M\|_F^2.
\end{aligned}$$

Next, we bound $\|AX - \tilde{A}X\|_F^2$. We have

$$\|AX - \tilde{A}X\|_F^2 = \|(D-\overline{D^{(t)}})V^\top X\|_F^2 \leq \|(D-\overline{D^{(t)}})\|_S^2 \|X\|_F^2 = r\sigma_{t+1}^2(A)$$

where the inequality follows because the spectral norm is consistent with the Euclidean norm. This implies the lemma since

$$r\sigma_{t+1}^2(A) \leq \epsilon(t-r+1)\sigma_{t+1}^2(A) \leq \epsilon \sum_{i=r+1}^{t+1} \sigma_i^2(A) \leq \epsilon \sum_{i=r+1}^{d} \sigma_i^2(A). \tag{40}$$

where the first inequality follows for our choice of t. □

B.3.2 Proof of Theorem 13

We first introduce some intermediate variables for analysis. Imagine we perform two projections: first project P_i to $\tilde{P}_i = P_i V_i^{(t)} (V_i^{(t)})^\top$, then project $\tilde{P}_i$ to $\overline{P}_i = \tilde{P}_i V^{(t)} (V^{(t)})^\top$. Let $\tilde{P}$ denote the vertical concatenation of $\tilde{P}_i$ and let $\overline{P}$ denote the vertical concatenation of $\overline{P}_i$, i.e.

$$\tilde{P} = \begin{bmatrix} \tilde{P}_1 \\ \vdots \\ \tilde{P}_s \end{bmatrix} \quad \text{and} \quad \overline{P} = \begin{bmatrix} \overline{P}_1 \\ \vdots \\ \overline{P}_s \end{bmatrix}$$

The following is a technical lemma that will be used in the proof of Theorem 13.

Lemma 27.

$$d^2(\tilde{P}, L(X)) \leq (1+\epsilon) d^2(P, L(X)).$$

Proof. We have

$$\begin{aligned} d^2(\tilde{P}, L(X)) - d^2(P, L(X)) &= \|\tilde{P} - \tilde{P}XX^\top\|_F^2 - \|P - PXX^\top\|_F^2 \\ &= \|\tilde{P}\|_F^2 - \|\tilde{P}XX^\top\|_F^2 - (\|P\|_F^2 - \|PXX^\top\|_F^2) \\ &= \sum_{i=1}^{s} \left[\|\tilde{P}_i\|_F^2 - \|P_i\|_F^2 \right] + \sum_{i=1}^{s} \left[\|P_i XX^\top\|_F^2 - \|\tilde{P}_i XX^\top\|_F^2 \right]. \end{aligned}$$

By the Pythagorean Theorem, $\|\tilde{P}_i\|_F^2 \leq \|P_i\|_F^2$. Also, since X is orthonormal, $\|P_i XX^\top\|_F^2 = \|P_i X\|_F^2$ and $\|\tilde{P}_i XX^\top\|_F^2 = \|\tilde{P}_i X\|_F^2$. Then

$$\begin{aligned} d^2(\tilde{P}, L(X)) - d^2(P, L(X)) &\leq \sum_{i=1}^{s} \left[\|P_i X\|_F^2 - \|\tilde{P}_i X\|_F^2 \right] \\ &\leq \sum_{i=1}^{s} \epsilon d^2(P_i, L(X)) = \epsilon d^2(P, L(X)) \end{aligned} \tag{41}$$

where the second inequality follows from Lemma 18. □

Theorem 13. *Let X be a $d \times j$ matrix whose columns are orthonormal. Let $\epsilon \in (0,1]$ and $t \in \mathbf{N}$ with $d-1 \geq t \geq j + \lceil 8j/\epsilon \rceil - 1$. Then the output of Algorithm 11 satisfies*

$$0 \leq \|PX - \hat{P}X\|_F^2 \leq \epsilon d^2(P, L(X)) \quad \text{and} \quad 0 \leq \|PX\|_F^2 - \|\hat{P}X\|_F^2 \leq \epsilon d^2(P, L(X)).$$

Proof. For the first statement, we have

$$\begin{aligned} \|PX - \hat{P}X\|_F^2 &\leq 2\|PX - \tilde{P}X\|_F^2 && (42) \\ &+ 2\|\tilde{P}X - \overline{P}X\|_F^2 && (43) \\ &+ 2\|\overline{P}X - \hat{P}X\|_F^2. && (44) \end{aligned}$$

For (42), we have by Lemma 18

$$\|PX - \tilde{P}X\|_F^2 = \sum_{i=1}^{s} \|P_iX - \tilde{P}_iX\|_F^2 \leq \sum_{i=1}^{s} \frac{\epsilon}{4} d^2(P_i, L(X)) = \frac{\epsilon}{8} d^2(P, L(X)). \quad (45)$$

Similarly, for (43) we have by Lemma 18

$$\|\tilde{P}X - \overline{P}X\|_F^2 \leq \frac{\epsilon}{8} d^2(\tilde{P}, L(X)). \quad (46)$$

To bound (44), let $Y = V^{(t)}(V^{(t)})^\top X$. Then by definition, $\overline{P}_iX = \tilde{P}_iY$ and $\hat{P}_iX = P_iY$. By Lemma 18, we have

$$\begin{aligned} \|\overline{P}X - \hat{P}X\|_F^2 &= \sum_{i=1}^{s} \|\tilde{P}_iY - P_iY\|_F^2 \leq \sum_{i=1}^{s} \frac{\epsilon}{8} \sum_{i=r+1}^{s} \sigma_i^2(P_i) \\ &\leq \frac{\epsilon}{8} \sum_{i=1}^{s} d^2(P_i, L(X)) = \frac{\epsilon}{8} d^2(P, L(X)). \quad (47) \end{aligned}$$

Combining (45)(46) and (47) leads to

$$\|PX - \hat{P}X\|_F^2 \leq \frac{\epsilon}{2} d^2(P, L(X)) + \frac{\epsilon}{4} d^2(\tilde{P}, L(X)). \quad (48)$$

The first statement then follows from (48) and Lemma 27.

For the second statement, we have a similar argument.

$$\begin{aligned} \|PX\|_F^2 - \|\hat{P}X\|_F^2 &= \|PX\|_F^2 - \|\tilde{P}X\|_F^2 && (49) \\ &+ \|\tilde{P}X\|_F^2 - \|\overline{P}X\|_F^2 && (50) \\ &+ \|\overline{P}X\|_F^2 - \|\hat{P}X\|_F^2. && (51) \end{aligned}$$

For (49), we have by Lemma 18

$$\begin{aligned} \|PX\|_F^2 - \|\tilde{P}X\|_F^2 &= \sum_{i=1}^{s} \left[\|P_i X\|_F^2 - \|\tilde{P}_i X\|_F^2 \right] && (52) \\ &\leq \sum_{i=1}^{s} \frac{\epsilon}{4} d^2(P_i, L(X)) = \frac{\epsilon}{4} d^2(P, L(X)). && (53) \end{aligned}$$

Similarly, for (50) we have by Lemma 18

$$\|\tilde{P}X\|_F^2 - \|\overline{P}X\|_F^2 \leq \frac{\epsilon}{4} d^2(\tilde{P}, L(X)). \tag{54}$$

By Lemma 18, we have

$$\begin{aligned} \|\overline{P}X\|_F^2 - \|\hat{P}X\|_F^2 &= \sum_{i=1}^{s} \left[\|\tilde{P}_i Y\|_F^2 - \|P_i Y\|_F^2 \right] \leq \sum_{i=1}^{s} \frac{\epsilon}{4} \sum_{i=r+1}^{s} \sigma_i^2(P_i) \\ &\leq \frac{\epsilon}{4} \sum_{i=1}^{s} d^2(P_i, L(X)) = \frac{\epsilon}{4} d^2(P, L(X)). && (55) \end{aligned}$$

Combining (53)(54) and (55) leads to

$$\|PX\|_F^2 - \|\hat{P}X\|_F^2 \leq \frac{\epsilon}{2} d^2(P, L(X)) + \frac{\epsilon}{4} d^2(\tilde{P}, L(X)). \tag{56}$$

The second statement then follows from (56) and Lemma 27. □

B.3.3 Proof of Theorem 14

The analysis follows the ideas in [46], but is tailored for the distributed setting. We first begin with the following lemma, showing that the cost of the projected data to any low dimension subspace approximates that of the original data, compensated by a positive constant.

Lemma 28. *Let X be a $d \times j$ matrix whose columns are orthonormal. Let $\epsilon \in (0, 1]$ and $t \in \mathbf{N}$ with $d - 1 \geq t \geq j + \lceil 4j/\epsilon \rceil - 1$. Then there exists $c_1 \geq 0$ such that*

$$0 \leq d^2(\hat{P}, L(X)) + c_1 - d^2(P, L(X)) \leq \epsilon d^2(P, L(X)).$$

Proof. We have from Pythagorean Theorem

$$\begin{aligned} d^2(\hat{P}, L(X)) - d^2(P, L(X)) &= ||\hat{P}||_2^2 - ||\hat{P}X||_2^2 - (||P||_2^2 - ||PX||_2^2) \\ &= ||PX||_2^2 - ||\hat{P}X||_2^2 - c_1 \end{aligned}$$

where $c_1 = ||P||_2^2 - ||\hat{P}||_2^2$. Note that by Fact **??**,

$$c_1 = \sum_{i=1}^{n} \left[||P_i||_2^2 - ||P_i^{(t)}||_2^2 \right] + ||P^{(t)}||_2^2 - ||\hat{P}||_2^2 \geq 0.$$

Then the lemma follows from Theorem 13. □

The next lemma shows that the projection of the projected data to any low dimension subspace approximates the projection of the projected data in the sense that their distances are small.

Let p_i denote the ith row of the data P, and let $\hat{p}_i$ denote the ith row of $\hat{P}$.

Lemma 29. *Let X be a $d \times j$ matrix whose columns are orthonormal. Let $\epsilon \in (0, 1]$ and $t \in \mathbf{N}$ with $d - 1 \geq t \geq j + \lceil 4j/\epsilon \rceil - 1$. Then*

$$\sum_{i=1}^{|P|} d(\Pi_X(p_i), \Pi_X(\hat{p}_i))^2 \leq \epsilon d^2(P, L(X)).$$

Proof. Since X is orthogonal, $\Pi_X(p) = pXX^T$. Then

$$\sum_{i=1}^{|P|} d(\Pi_X(p_i), \Pi_X(\hat{p}_i))^2 = \sum_{i=1}^{|P|} ||p_i XX^T - \hat{p}_i XX^T||_2^2 = ||PXX^T - \hat{P}XX^T||_2^2.$$

This can be simplified to $||(P - \hat{P})X||_2^2$ since

$$\begin{aligned} ||PXX^T - \hat{P}XX^T||_2^2 &= ||(P - \hat{P})XX^T||_2^2 = \text{trace}[(P - \hat{P})XX^T XX^T (P - \hat{P})^T)] \\ &= \text{trace}[(P - \hat{P})XX^T (P - \hat{P})^T)] = ||(P - \hat{P})X||_2^2. \end{aligned}$$

The lemma then follows from Theorem 13. □

The above two lemmas are the key elements needed to show our final theorem. Before proving the theorem, we further need the following "weak triangle inequality", which is well known in the coreset literature. The proof is included in the appendix for completeness.

Lemma 30. *[Lemma 7.1 in [46]] For any $0 \leq \epsilon \leq 1$, a compact set $C \subseteq \mathbf{R}^d$, and $p, q \in \mathbf{R}^d$,*

$$|d(p,C)^2 - d(q,C)^2| \leq \frac{12d(p,q)^2}{\epsilon} + \frac{\epsilon}{2}d(p,C)^2.$$

Proof. Using the triangle inequality,

$$\begin{aligned} d(p,C)^2 - d(q,C)^2| &= |d(p,C) - d(q,C)| \cdot (d(p,C) + d(q,C)) \\ &\leq d(p,q)(2d(p,C) + d(p,q)) \\ &\leq d(p,q)^2 + 2d(p,C)d(p,q). \end{aligned} \tag{57}$$

Either $d(p,C) \leq d(p,q)/\epsilon$ or $d(p,q) < \epsilon d(p,C)$. Hence,

$$d(p,C)d(p,q) \leq \frac{d(p,q)^2}{\epsilon} + \epsilon d(p,C)^2.$$

Combining the last inequality with (57) yields

$$|d(p,C)^2 - d(q,C)^2| \leq d(p,q)^2 + \frac{2d(p,q)^2}{\epsilon} + 2\epsilon d(p,C)^2 \leq \frac{3d(p,q)^2}{\epsilon} + 2\epsilon d(p,C)^2.$$

Finally, the lemma follows by replacing ϵ with $\epsilon/4$. □

We are now ready to prove the theorem, which guarantees that a coreset for the output of the distributed PCA algorithm is also a coreset for the original data.

Theorem 14. *Let $\mathbf{x}$ be a set of k centers in $\mathbf{R}^d$. Let $\epsilon \in (0,1]$ and $t \in \mathbf{N}$ with $d - 1 \geq t \geq k + \lceil 50k/\epsilon^2 \rceil$. Then there exists a constant $c_0 \geq 0$, such that the output of Algorithm 11 satisfies*

$$(1-\epsilon)d^2(P,\mathbf{x}) \leq d^2(\hat{P},\mathbf{x}) + c_0 \leq (1+\epsilon)d^2(P,\mathbf{x}).$$

Proof. Let $X \in \mathbf{R}^{d \times k}$ has orthonormal columns that span $\mathbf{x}$. Let c_0 be the constant c_1 in Lemma 28. Then by Pythagorean theorem we have

$$\begin{aligned} & |d^2(\hat{P},\mathbf{x}) + c_0 - d^2(P,\mathbf{x})| \\ = & \left| d^2(\hat{P}, L(X)) + c_0 - d^2(P, L(X)) + \sum_{i=1}^{|P|} \left[d(p_X(p_i),\mathbf{x})^2 - d(p_X(\hat{p}_i),\mathbf{x})^2 \right] \right|. \end{aligned}$$

By Lemma 28 we have

$$\left| d^2(\hat{P}, L(X)) + c_0 - d^2(P, L(X)) \right| \leq \frac{\epsilon^2}{4} d^2(P, L(X)). \tag{58}$$

By Lemma 29 and Lemma 30 we have

$$\begin{aligned} \sum_{i=1}^{|P|} \left| d(p_X(p_i), \mathbf{x})^2 - d(p_X(\hat{p}_i), \mathbf{x})^2 \right| &\leq \sum_{i=1}^{|P|} \left[\frac{12 d(p_X(p_i), p_X(\hat{p}_i))^2}{\epsilon} + \frac{\epsilon}{2} d(p_X(p_i), \mathbf{x})^2 \right] \\ &\leq \frac{\epsilon}{4} d^2(P, \mathbf{x}) + \frac{\epsilon}{2} \sum_{i=1}^{|P|} d(p_X(p_i), \mathbf{x})^2. \end{aligned} \tag{59}$$

Since $d^2(P, L(X)) \leq d^2(P, \mathbf{x})$ and $d(p_X(p_i), \mathbf{x}) \leq d(p_i, \mathbf{x})$, the theorem follows from (58)(59). □

REFERENCES

[1] Ackerman, M. and Ben-David, S., "Measures of clustering quality: A working set of axioms for clustering," in *Advances in Neural Information Processing Systems*, 2008.

[2] Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P., "Mixed membership stochastic blockmodels," *The Journal of Machine Learning Research*, 2008.

[3] Albert, R. and Barabási, A.-L., "Statistical mechanics of complex networks," *Reviews of Modern Physics*, 2002.

[4] Arora, S., Ge, R., Sachdeva, S., and Schoenebeck, G., "Finding overlapping communities in social networks: toward a rigorous approach," in *Proceedings of the ACM Conference on Electronic Commerce*, 2012.

[5] Arthur, D. and Vassilvitskii, S., "k-means++: The advantages of careful seeding," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[6] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V., "Local search heuristics for k-median and facility location problems," *SIAM Journal of Computing*, 2004.

[7] Awasthi, P. and Balcan, M., "Center based clustering: A foundational perspective." Survey Chapter in Handbook of Cluster Analysis (Manuscript), 2013.

[8] Awasthi, P., Blum, A., and Sheffet, O., "Stability yields a ptas for k-median and k-means clustering," in *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2010.

[9] Awasthi, P., Blum, A., and Sheffet, O., "Center-based clustering under perturbation stability," *Information Processing Letters*, 2012.

[10] Awasthi, P. and Sheffet, O., "Improved spectral-norm bounds for clustering," in *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, 2012.

[11] Bache, K. and Lichman, M., "UCI machine learning repository," 2013.

[12] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S., "Scalable k-means++," in *Proceedings of the International Conference on Very Large Data Bases*, 2012.

[13] Bai, Z.-J., Chan, R. H., and Luk, F. T., "Principal component analysis for distributed data sets with updating," in *Advanced Parallel Processing Technologies*, 2005.

[14] Balcan, M. F., Blum, A., and Gupta, A., "Approximate clustering without the approximation," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[15] Balcan, M. F. and Braverman, M., "Approximate nash equilibria under stability conditions," *CoRR*, vol. abs/1008.1827, 2010.

[16] Balcan, M. F. and Gupta, P., "Robust hierarchical clustering," in *Proceedings of the Annual Conference on Learning Theory*, 2010.

[17] Balcan, M.-F., Berlind, C., Ehrlich, S., and Liang, Y., "Efficient semi-supervised and active learning of disjunctions," in *Proceedings of the International Conference on Machine Learning*, 2013.

[18] Balcan, M.-F., Blum, A., Fine, S., and Mansour, Y., "Distributed learning, communication complexity and privacy," in *Proceedings of the Conference on Learning Thoery*, 2012.

[19] Balcan, M. F. and Liang, Y., "Clustering under perturbation resilience," in *Proceedings of the International Colloquium on Automata, Languages, and Programming*, 2012.

[20] Balcan, M. F. and Liang, Y., "Modeling and detecting community hierarchies," in *Proceedings of the International Workshop on Similarity-Based Pattern Analysis and Recognition*, 2013.

[21] Balcan, M., Borgs, C., Braverman, M., Chayes, J., and Teng, S., "Finding endogenously formed communities," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[22] Bandelt, H. and Dress, A., "Weak hierarchies associated with similarity measures – an additive clustering technique," *Bulletin of Mathematical Biology*, 1989.

[23] Bartal, Y., Charikar, M., and Raz, D., "Approximating min-sum k-clustering in metric spaces," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2001.

[24] Bellet, A., Liang, Y., Garakani, A. B., Balcan, M.-F., and Sha, F., "Distributed frank-wolfe algorithm: A unified framework for communication-efficient sparse learning," *CoRR*, vol. abs/1404.2644, 2014.

[25] Ben-David, S., "A framework for statistical clustering with a constant time approximation algorithms for k-median clustering," *Proceedings of Annual Conference on Learning Theory*, 2004.

[26] Bilu, Y. and Linial, N., "Are stable instances easy?," in *Proceedings of the Symposium on Innovations in Computer Science*, 2010.

[27] Bilu, Y., Daniely, A., Linial, N., and Saks, M., "On the practically interesting instances of maxcut," in *Proceedings of the International Symposium on Theoretical Aspects of Computer Science*, 2013.

[28] Bounova, G. and de Weck, O., "Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles," *Physical Review E*, 2012.

[29] Charikar, M., Guha, S., Tardos, É., and Shmoys, D. B., "A constant-factor approximation algorithm for the k-median problem," *Journal of Computer and System Sciences*, 2002.

[30] Chen, K., "On k-median clustering in high dimensions," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[31] Cheng, D., Kannan, R., Vempala, S., and Wang, G., "A divide-and-merge methodology for clustering," *ACM Transactions on Database Systems*, vol. 31, no. 4, 2006.

[32] Clauset, A., Moore, C., and Newman, M., "Hierarchical structure and the prediction of missing links in networks," *Nature*, 2008.

[33] Clauset, A., Newman, M. E. J., and Moore, C., "Finding community structure in very large networks," *Physical Review E*, 2004.

[34] Condon, A. and Karp, R. M., "Algorithms for graph partitioning on the planted partition model," *Random Structures and Algorithms*, vol. 18, no. 2, 2001.

[35] Considine, J., Li, F., Kollios, G., and Byers, J., "Approximate aggregation techniques for sensor databases," in *Proceedings of the International Conference on Data Engineering*, 2004.

[36] Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., and others, "Spanner: Googles globally-distributed database," in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, 2012.

[37] Dasgupta, S., "Learning mixtures of gaussians," in *Proceedings of the Annual Symposium on Foundations of Computer Science*, 1999.

[38] Datta, S., Giannella, C., Kargupta, H., and others, "K-means clustering over peer-to-peer networks," in *Proceedings of the International Workshop on High Performance and Distributed Mining*, 2005.

[39] Daumé III, H., Phillips, J. M., Saha, A., and Venkatasubramanian, S., "Efficient protocols for distributed classification and optimization," in *Algorithmic Learning Theory*, pp. 154–168, Springer, 2012.

[40] de la Vega, W. F., Karpinski, M., Kenyon, C., and Rabani, Y., "Approximation schemes for clustering problems," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2003.

[41] Ding, C. and He, X., "K-means clustering via principal component analysis," in *Proceedings of the International Conference on Machine learning*, 2004.

[42] Du, N., Liang, Y., Balcan, M.-F., and Song, L., "Budgeted influence maximization for multiple products," *CoRR*, vol. abs/1312.2164, 2013.

[43] Du, N., Liang, Y., Balcan, M.-F., and Song, L., "Influence function learning in information diffusion networks," 2014.

[44] Everitt, B. S., Landau, S., Leese, M., and Stahl, D., *Hierarchical Clustering*. 2011.

[45] Feldman, D. and Langberg, M., "A unified framework for approximating and clustering data," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2011.

[46] Feldman, D., Schmidt, M., and Sohler, C., "Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[47] Feldman, D., Sugaya, A., and Rus, D., "An effective coreset compression algorithm for large scale sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2012.

[48] Forman, G. and Zhang, B., "Distributed data clustering can be efficient and exact," *ACM SIGKDD Explorations Newsletter*, 2000.

[49] Fortunato, S., "Community detection in graphs," *Physics Reports*, 2010.

[50] Ghashami, M. and Phillips, J., "Relative errors for deterministic low-rank matrix approximations," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014.

[51] Girvan, M. and Newman, M. E. J., "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, 2002.

[52] Girvan, M. and Newman, M. E. J., "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, 2002.

[53] Greenhill, S. and Venkatesh, S., "Distributed query processing for mobile surveillance," in *Proceedings of the International Conference on Multimedia*, 2007.

[54] Greenwald, M. and Khanna, S., "Power-conserving computation of order-statistics over sensor networks," in *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2004.

[55] Guha, S. and Khuller, S., "Greedy strikes back: Improved facility location algorithms," *Journal of Algorithms*, 1999.

[56] Har-Peled, S. and Kushal, A., "Smaller coresets for k-median and k-means clustering," *Discrete & Computational Geometry*, 2007.

[57] Har-Peled, S. and Mazumdar, S., "On coresets for k-means and k-median clustering," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.

[58] He, J., Hopcroft, J., Liang, H., Suwajanakorn, S., and Wang, L., "Detecting the structure of social networks using (α, β)-communities," in *Proceedings of the International Conference on Algorithms and Models for the Web Graph*, 2011.

[59] Jain, K., Mahdian, M., and Saberi, A., "A new greedy approach for facility location problems," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2002.

[60] Januzaj, E., Kriegel, H., and Pfeifle, M., "Towards effective and efficient distributed clustering," in *Workshop on Clustering Large Data Sets in the IEEE International Conference on Data Mining*, 2003.

[61] Kannan, R., Vempala, S., and Vetta, A., "On clusterings: Good, bad and spectral," *Journal of the ACM*, vol. 51, no. 3, 2004.

[62] Kannan, R. and Vempala, S., "Nimble algorithms for cloud computing," *CoRR*, vol. abs/1304.3162, 2013.

[63] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y., "A local search approximation algorithm for k-means clustering," in *Proceedings of the Annual Symposium on Computational Geometry*, 2002.

[64] Kargupta, H., Huang, W., Sivakumar, K., and Johnson, E., "Distributed clustering using collective principal component analysis," *Knowledge and Information Systems*, 2001.

[65] Kim, M. and Leskovec, J., "Latent multi-group membership graph model," in *Proceedings of the International Conference on Machine Learning*, 2012.

[66] King, B., "Step-wise clustering procedures," *Journal of the American Statistical Association*, 1967.

[67] Kleinberg, J., "An impossibility theorem for clustering," in *Advances in Neural Information Processing Systems*, 2002.

[68] Krebs, V., "http://www.orgnet.com/," *unpublished*.

[69] Kumar, A. and Kannan, R., "Clustering with spectral norm and the k-means algorithm," in *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2010.

[70] Kumar, A. and Kannan, R., "Clustering with spectral norm and the k-means algorithm," in *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[71] Lagomarsino, M. C., Jona, P., Bassetti, B., and Isambert, H., "Hierarchy and feedback in the evolution of the Escherichia coli transcription network," *Proceedings of the National Academy of Sciences*, 2007.

[72] Lancichinetti, A. and Fortunato, S., "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, 2009.

[73] Lancichinetti, A., Fortunato, S., and Kertész, J., "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, 2009.

[74] Langberg, M. and Schulman, L., "Universal ε-approximators for integrals," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[75] Le Borgne, Y.-A., Raybaud, S., and Bontempi, G., "Distributed principal component analysis for wireless sensor networks," *Sensors*, 2008.

[76] Li, S. and Svensson, O., "Approximating k-median via pseudo-approximation," in *Proceedings of the ACM Symposium on the Theory of Computing*, 2013.

[77] Li, Y., Long, P. M., and Srinivasan, A., "Improved bounds on the sample complexity of learning," in *Proceedings of the eleventh annual ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[78] Liang, Y., Balcan, M.-F., and Ehrlich, S., "Distributed k-means and k-median clustering on general topologies," in *Advances in Neural Information Processing Systems*, 2013.

[79] Liang, Y., Balcan, M.-F., and Kanchanapally, V., "Distributed pca and k-means clustering," in *The Big Learning Workshop, Advances in Neural Information Processing Systems*, 2013.

[80] Lipton, R. J., Markakis, E., and Mehta, A., "On stability properties of economic solution concepts." Manuscript, 2006.

[81] Lloyd, S., "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, 1982.

[82] Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., and Dawson, S. M. *Behavioral Ecology and Sociobiology*, 2003.

[83] Macua, S. V., Belanovic, P., and Zazo, S., "Consensus-based distributed principal component analysis in wireless sensor networks," in *Proceedings of the IEEE International Workshop on Signal Processing Advances in Wireless Communications*, 2010.

[84] Makarychev, K., Makarychev, Y., and Vijayaraghavan, A., "Bilulinial stable instances of max cut," *CoRR*, vol. abs/1305.1681, 2013.

[85] Matula, D. W. and Shahrokhi, F., "Sparsest cuts and bottlenecks in graphs," *Discrete Applied Mathematics*, vol. 27, no. 1, 1990.

[86] Mihalák, M., Schöngens, M., Šrámek, R., and Widmayer, P., "On the complexity of the metric tsp under stability considerations," in *Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science*, 2011.

[87] Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D., and Mahanti, A., "Characterizing web-based video sharing workloads," *ACM Transactions on the Web*, 2011.

[88] Newman, M. E. J., "Detecting community structure in networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, 2004.

[89] Newman, M. E. J., "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, 2006.

[90] Newman, M. E. and Girvan, M., "Mixing patterns and community structure in networks," in *Statistical mechanics of complex networks*, 2003.

[91] Olston, C., Jiang, J., and Widom, J., "Adaptive filters for continuous queries over distributed data streams," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[92] Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C., "The effectiveness of lloyd-type methods for the k-means problem," in *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, 2006.

[93] Qu, Y., Ostrouchov, G., Samatova, N., and Geist, A., "Principal component analysis for dimension reduction in massive distributed data sets," in *Proceedings of IEEE International Conference on Data Mining*, 2002.

[94] Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., and Barabási, A. L., "Hierarchical Organization of Modularity in Metabolic Networks," *Science*, 2002.

[95] Reyzin, L., "Data stability in clustering: A closer look," in *Proceedings of the International Conference on Algorithmic Learning Theory*, 2012.

[96] Schölkopf, B. and Smola, A. J., *Learning with Kernels*. 2002.

[97] Schweinberger, M. and Snijders, T. A. B., "Settings in social networks: A measurement model," *Sociological Methodology*, 2003.

[98] Sneath, P. H. A. and Sokal, R. R., *Numerical taxonomy. The principles and practice of numerical classification.* 1973.

[99] Spielman, D. A. and Teng, S.-H., "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.

[100] Steinbach, M., Karypis, G., Kumar, V., and others, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 400, 2000.

[101] Tasoulis, D. and Vrahatis, M., "Unsupervised distributed clustering," in *Proceedings of the International Conference on Parallel and Distributed Computing and Networks*, 2004.

[102] Vapnik, V. N., *Statistical learning theory*. 1998.

[103] Zachary, W. W., "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, 1977.

[104] Zhang, Q., Liu, J., and Wang, W., "Approximate clustering on distributed data streams," in *Proceedings of the IEEE International Conference on Data Engineering*, 2008.

INDEX

VITA

Yingyu Liang was born in Pingnan, Guangxi Province of China. After completing his schoolwork at Guigang High School in Guangxi in 2004, Yingyu entered Tsinghua University in Beijing. He received a Bachelor of Science in July 2008 and a Master of Science in July 2010 with a major in computer science and technology from Tsinghua University. In August 2010, he entered the School of Computer Science in Georgia Institute of Technology in Atlanta, Georgia.