The Dissertation Committee for Ramgopal Reddy Mettu
certifies that this is the approved version of the following dissertation:

Approximation Algorithms for $\mathcal{NP}$-Hard Clustering Problems

Committee:

C. Greg Plaxton, Supervisor

Annamaria Nina Amenta

Inderjit Dhillon

Anna Gál

Joydeep Ghosh

Approximation Algorithms for $\mathcal{NP}$-Hard Clustering Problems

by

Ramgopal Reddy Mettu, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2002

Acknowledgments

During my graduate career, there were many individuals that helped me become a better researcher as well as a better person. First and foremost, I would like to express my gratitude to my advisor Greg Plaxton for his valuable guidance and boundless enthusiasm. His attitude toward problem solving and research in general have been truly enlightening, and I consider myself lucky for having had the opportunity to work with him for the last few years. I can remember many marathon research meetings that were fueled by Greg's ideas and enthusiasm. Whether he was in Austin or on a leave of absence several thousand miles away, Greg was always available and willing to contribute. All of the results in this dissertation were co-authored with Greg. I am also deeply indebted to my family for providing gentle, but firm, guidance through the times when I was unsure of where I was headed. Antonio Garza and Aaron Walker have been two of my best friends for much of my adult life. Their decidedly unscientific influence, especially over the last five years, has left an indelible mark on my world-view and attitude toward life. During his time here in Austin, and even afterward, Antonio has been an everyday friend and kindred spirit. There has been many a discussion when our mutual give-and-take led us in interesting and illuminating directions. Aaron's razor-sharp intellect and brilliant ability to reflect have helped me realize that rigorous thought is a beautiful thing even outside of the sciences. Umut Acar has remained a steadfast friend even though he was in Austin for just a short while. Umut not only gave me a trial-by-fire introduction to rock climbing, but was also a great roommate and sounding board for ideas. Cycling would not have been nearly as painful, or rewarding, without a riding partner like Eric Daniels. Eric is a genuine friend for whom generosity is never lacking. Emery Berger, Brendon Cahoon, Sam Guyer, Daniel Jiménez, and Phoebe Weidemann were certainly the best group of lunch companions and systems folks that I could have hoped for. Our lunches could have been more efficient and our practice talks shorter, but they gave us a reason, and time, to learn about and critique each other's research. I have to thank Emery especially; even while he was working on one of his many research ideas, he always took time to answer my questions about miscellaneous Microsoft products. And last but certainly not least, Rajesh Cherian has been a close friend whose motorcycle knowledge and magnanimity were invaluable.

It is typical to acknowledge institutional support, and while NSF funding[1] made my research possible, there have been other, smaller institutions that were just as important to my well-being throughout my many years in Austin. Little City arguably has the best espresso, and has always facilitated relaxing coffee breaks after research meetings. Vulcan Video has been an invaluable source of art and foreign films, many of which helped me exercise my right brain. Although I haven't yet taken advantage of "happy minutes" at the Texas Showdown Saloon, there were many relaxing bull sessions that took place over a pitcher at the Showdown. Finally, I'd like to thank Nelo's Pro Cycles for the Wednesday night rides that were, brief, and painful but welcome respites.

Ramgopal Reddy Mettu

The University of Texas at Austin

August 2002

[1]The work in this dissertation was supported by NSF Grant CCR–9821053.

Approximation Algorithms for $\mathcal{NP}$-Hard Clustering Problems

Publication No. ____________________

Ramgopal Reddy Mettu, Ph.D.
The University of Texas at Austin, 2002

Supervisor: C. Greg Plaxton

Given a set of n points and their pairwise distances, the goal of clustering is to partition the points into a "small" number of "related" sets. Clustering algorithms are used widely to manage, classify, and summarize many kinds of data. In this dissertation, we study the classic *facility location* and *k-median* problems in the context of clustering, and formulate and study a new optimization problem that we call the *online median* problem. For each of these problems, it is known to be $\mathcal{NP}$-hard to compute a solution with cost less than a certain constant factor times the optimal cost. We give simple constant-factor approximation algorithms for the facility location, k-median, and online median problems with optimal or near-optimal time bounds. We also study distance functions that are "approximately" metric, and show that such distance functions allow us to obtain a faster online median algorithm and to generalize our analysis to other objective functions, such as that of the well-known k-means heuristic.

Given n points, the associated interpoint distances and nonnegative point weights, and a nonnegative penalty for each point, the facility location problem asks us to identify a set of cluster centers so that the weighted average cluster radii and the sum of the cluster center penalties are both minimized. The k-median problem asks us to identify exactly k cluster centers while minimizing just the weighted average cluster radii. We give a simple greedy algorithm for the facility location problem that runs in $O(n^2)$ time and produces a solution with cost at most 3 times optimal. For the k-median problem, we develop and make use of a sampling technique that we call *successive sampling*, and give a randomized constant-factor approximation algorithm that runs in $O(n(k + \log n + \log^2 n))$ time. We also give an $\Omega(nk)$ lower bound on the running time of any randomized constant-factor approximation algorithm for the k-median problem that succeeds with even a negligible constant probability.

In many settings, it is desirable to browse a given data set at differing levels of granularity (i.e., number of clusters). To address this concern, we formulate a generalization of the k-median problem that we call the *online median problem*. The online median problem asks us to compute an ordering of the points so that, over all i, when a prefix of length i is taken as a set of cluster centers, the weighted average radii of the induced clusters is minimized. We show that a natural generalization of the greedy strategy that we call *hierarchically greedy* yields an algorithm that produces an ordering such that every prefix of the ordering is within a constant factor of the associated optimal cost. Furthermore, our algorithm has a running time of $\Theta(n^2)$.

Finally, we study the performance of our algorithms in practice. We present implementations of our k-median and online median algorithms; our experimental results indicate that our approximation algorithms may be useful in practice.

Table of Contents

List of Figures

Chapter 1

Introduction

Given a set of n weighted points and their pairwise distances, the goal of clustering is to partition the points into a "small" number of "related" sets. Clustering algorithms are used widely to manage, classify, and summarize many kinds of data. For example, Data Mining applications rely heavily on efficient clustering algorithms to infer structure in massive data sets. We study two classic discrete location problems, *facility location* and *k-median* problems, in the context of clustering. We also formulate and study a generalization of the k-median problem that we call the *online median problem*. We demonstrate that it is possible to obtain simple constant-factor approximation algorithms for these problems with optimal or near-optimal time bounds. We also study distance functions that are "approximately" metric, and show that such distance functions allow us to obtain a faster online median algorithm and to generalize our analysis to other objective functions, such as that of the well-known k-means heuristic.

As the name may indicate, the facility location problem was originally studied in operations research and is perhaps the canonical problem in *discrete location theory* [33, 38]. Given a set of cities and the intercity distances, suppose that we wish to open a number of facilities to provide some resource to the cities. Suppose also that each city also has a cost associated with opening a facility in that city. Then the facility location problem asks us to identify a subset of the cities in which to open facilities while minimizing the total opening cost as well as the sum, over all cities, of the distance to the closest facility. This problem is useful for the problem of resource placement, but serves equally well to model the problem of clustering. To see this, we simply view the cities as objects that we wish to cluster (e.g., documents), the opening costs as object penalties (e.g., infrequency of access) and the distances as similarities between objects (e.g., number of words in common). Then the facility location problem is a natural formulation of the clustering problem since it asks us to identify a set of cluster centers so that the weighted average cluster radii and the sum of the cluster center penalties are both minimized.

Another classic problem in operations research that corresponds to a well-studied clustering problem is the k-median problem [38]. The k-median problem asks to identify exactly k cluster centers while minimizing just the weighted average cluster radii. We note that this problem is very similar to k-means clustering [9]. In many settings, it is desirable to browse a given data set at differing levels of granularity (i.e., number of clusters). To address this concern, we formulate a generalization of the k-median problem that we call the *online median problem.* The online median problem asks us to compute an ordering of the points so that, over all i, when a prefix of length i is taken as a set of cluster centers, the weighted average radii of the clusters is minimized.

This dissertation focuses on developing approximation algorithms for these problems since, for each of them, it is $\mathcal{NP}$-hard to compute a solution with cost less than a certain constant factor times the optimal cost. We give a simple greedy algorithm for the facility location problem that runs in $O(n^2)$ time and produces a solution with cost at most 3 times optimal. In the course of developing algorithms for the k-median and online median problems, we introduce two new algorithmic techniques that may be of independent interest. For the k-median problem, we develop and make use of a sampling technique that we call *successive sampling*, and give a randomized constant-factor approximation algorithm that runs in $O(n(k + \log n + \log^2 n))$ time. We also give an $\Omega(nk)$ lower bound on the running time of any randomized constant-factor approximation algorithm for the k-median problem that succeeds with even a negligible constant probability. For the online median problem, we investigate whether it is possible to compute an ordering of the points with the property that every prefix of length i of the ordering has cost within a constant factor of optimal. That is, for every prefix of length i, when taken as a set of cluster centers, has cost within a constant factor of the cost of the optimal i cluster centers. We show that a natural generalization of the greedy strategy that we call *hierarchically greedy* yields an algorithm that produces an ordering such that every prefix of the ordering is within a constant factor of the associated optimal cost. Furthermore, our algorithm runs in $O(n^2)$ time and is thus time-optimal.

We also study the performance of our algorithms in practice. We present implementations of our k-median and online median algorithms and study their performance. Our results indicate that our approximation algorithms may be useful in practice. For synthetically generated inputs, our k-median algorithm provides a reasonable tradeoff between solution quality and speed when compared to a typical implementation of the widely used k-means heuristic. We also provide anecdotal evidence that our online median algorithm may be useful for real-world applications.

1.1 Problem Definitions

We first introduce some useful terminology that we use throughout this dissertation. Fix a set of points U, a distance function $d : U \times U \rightarrow \mathbb{R}$, and nonnegative functions $f, w : U \rightarrow \mathbb{R}$. For the purposes of this section, we assume that d is a metric, that is, d is nonnegative, symmetric, satisfies the triangle inequality, and $d(x, y) = 0$ iff

$x = y$. We let $n = |U|$, and define a subset of U to be a ***configuration*** iff it is nonempty. (Remark: A configuration is simply a set of cluster centers.) For any point x and configuration X, we define $d(x, X)$ as $\min_{y \in X} d(x, y)$. For any set of points X, we let $w(X)$ denote $\sum_{x \in X} w(x)$.

We consider three computational problems: facility location, k-median, online median. For the facility location problem, the ***cost*** of a configuration, denoted *cost* (X), is defined as the sum of $\sum_{x \in X} f(x)$ and $\sum_{x \in U} d(x, X) \cdot w(x)$. The input to the facility location problem is (U, d), f, and w. The output is a minimum-cost configuration. For the k-median and online median problems, the ***cost*** of a configuration, which we denote as *cost* (X), is defined to be $\sum_{x \in U} d(x, X) \cdot w(x)$. The input to the k-median problem is (U, d), w, and an integer k, $0 < k \leq n$. The output is a minimum-cost configuration of size k. The input to the online median problem is (U, d) and w. The output is a total order on U. We define the competitive ratio of such an ordering as the maximum over all k, $0 < k \leq n$, of the ratio of the cost of the configuration given by the first k points in the ordering to that of an optimal k-median configuration. We define the ***competitive ratio*** of an online median algorithm as the supremum, over all possible choices of the input instance (U, d) and w, of the competitive ratio of the ordering produced by the algorithm. An online median algorithm that guarantees a ratio of at most r is said to achieve a *competitive ratio* of r, or to be r-*competitive*.

1.2 Our Techniques and Contributions

Algorithms for problems in discrete location theory arise in many practical applications (see, e.g., [33, 38], for numerous pointers to the literature). Given that many of these problems are $\mathcal{NP}$-hard, it is desirable to develop fast approximation algorithms. For the k-median problem, we obtain an algorithm that is time optimal for most values of k and with high probability produces a solution whose cost is within a constant factor of optimal. Our algorithm is based on a sampling technique that we call *successive sampling* that may be of independent interest. For the facility location and online median problems, we show that greedy strategies yield simple and fast approximation algorithms. While a very simple greedy algorithm yields a constant-factor approximation bound for the facility location problem, it appears that a more sophisticated approach is needed to obtain a constant-factor approximation guarantee for the k-median problem, let alone a constant-competitiveness result for the online median problem. For example, in Section 5.1 we show that perhaps the most natural greedy approach to the k-median (resp., online median) problem leads to an unbounded approximation (resp., competitive) ratio. Our approach to the online median problem is based on a generalization of the greedy strategy that we call the *hierarchically greedy strategy.* We also show that all of the approximation results in this dissertation generalize to distance functions that are only "approximately" metric.

1.2.1 Successive Sampling

A natural technique to cope with a large set of unlabeled data is to take a random sample of the input in the hopes of capturing the essence of the input and subsituting the sample for the original input. Ideally we hope that the sample size required to capture the relevant information in the input is significantly less than the original input size. However, in many situations naive sampling does not always yield the desired reduction in data. For example, for the problem of learning Gaussians, this limitation manifests itself in the common assumption that the mixing weights are large enough so that a random sample of the data will capture a nonnegligible amount of the mass in a given Gaussian. Without this assumption, the approximation guarantees of recent algorithms for learning Gaussians [2, 8] no longer hold.

A major contribution of our work is a simple yet powerful sampling technique that we call *successive sampling*. We show that our sampling technique is an effective data reduction technique for the purpose of clustering in the sense it captures the essence of the input with a very small subset (just $O(k \log(n/k))$, where k is the number of clusters) of the points. In fact, it is this property of our sampling technique that allows us to develop an algorithm for the k-median problem that has an optimal running time of $\Theta(nk)$ for k between $\log n$ and $n/\log^2 n$ and that, with high probability, produces a solution with cost within a constant factor of optimal.

1.2.2 A Hierarchically Greedy Strategy

An obvious approach to the online median problem is to iteratively choose the point that minimizes the objective function. Greedy strategies of this kind are commonly applied in the design of online algorithms [4, 20] as well as approximation algorithms [18] (for a more detailed discussion see Chapter 2). It turns out, however, that for the online median problem, the simple greedy strategy mentioned above has an unbounded competitive ratio. We consider a generalization of this strategy that we call *hierarchically greedy*. The basic idea behind the hierarchically greedy strategy is as follows: Rather than selecting the next point in the ordering based on a single greedy criterion, we greedily choose a region (the set of points lying within some ball) and then recursively select a point within that region. Thus, the choice of point is influenced by a sequence of greedy criteria addressing successively finer levels of granularity. We show that the hierarchically greedy strategy yields a constant-competitive algorithm for the online median problem. Furthermore, our online median algorithm has an optimal running time of $\Theta(n^2)$.

1.2.3 Approximate Metrics

We also show that our analysis holds for classes of distance functions that are more general than metrics. We study two classes of "approximate" metrics for which the triangle inequality only holds to within a constant factor. We define and study λ-*approximate metrics* and *weakly* λ-*approximate metrics*. We show that our analysis holds to within constant factors given either of these two classes of distance functions. First, we show that λ-approximate distance functions facilitate an implementation of our online median algorithm running in time linear in the input size. We then show that weakly λ-approximate distance functions allow us to apply our techniques to objective functions other than the k-median objective. For example, we show that the approximation bounds for both of our algorithms hold to within constant factors for the well-known k-means objective function [9].

1.3 Summary of Results

We give an algorithm for the facility location problem that achieves an approximation ratio of 3 and runs in $\Theta(n^2)$ time. The main idea of the algorithm is to compute and use the "value" of balls about every point in the metric space. In retrospect, the idea of value is implicit in the work of Jain and Vazirani [22]. We make this idea explicit and use the values of balls to make greedy choices. Additionally, our algorithm is faster than the Jain-Vazirani algorithm by a logarithmic factor.

Under the standard assumption that the point weights and interpoint distances are polynomially bounded, we obtain a randomized $O(1)$-approximate algorithm for the k-median problem that runs in $O(n(k + \log n) + k^2 \log^2 n)$ time. For wide range of values of k, namely $\log n \leq k \leq n/\log^2 n$, this time bound simplifies to $O(nk)$. We also establish a matching $\Omega(nk)$ lower bound on the running time of any randomized $O(1)$-approximate k-median algorithm with a nonnegligible success probability (e.g., at least $\frac{1}{100}$). It is worth noting that in our study of the k-median problem, we present our results with respect to two additional parameters that capture the magnitude of the distances and point weights, respectively.

For the online median problem, we give a deterministic constant-competitive algorithm for the online median problem running in $O(n^2 + \ell n)$ time (where ℓ is the number of bits required to represent each distance). We note that our lower bound for the k-median problem implies that any constant-competitive online median algorithm requires $\Omega(n^2)$ time. Thus, under the standard assumption that $\ell = O(n)$, we achieve a time bound of $\Theta(n^2)$ for the online median problem. While we formulate and solve the online median problem, it worth noting that the k-median problem is a special case of the online median problem. Hence our online median algorithm is also the first deterministic constant-factor approximation algorithm for the k-median problem running in time that is linear in the size of the input. (The best previous deterministic running time of $O((n^2 \log n)(\ell + \log n))$ is due to Jain and Vazirani [22].)

We also show that all of our approximation results generalize to certain classes of distance functions that are only "approximately" metric. We also show that these generalizations are sufficient to consider objective functions other than that of the facility location, k-median and online median problems. For example, we show that our approximation results hold for the well-known k-means objective function.

1.4 Outline of Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we give a brief survey of previous approaches to clustering as well as the facility location results that are most relevant to our work. In Chapter 3, we present our $O(n^2)$-time constant-factor approximation algorithm for the facility location problem. In Chapter 4, we present a constant-factor approximation algorithm for the k-median problem and show that for a wide range of values of k, our algorithm runs in $O(nk)$ time. We also show that any randomized constant-factor approximation algorithm for the k-median problem with even a negligible success probability (say $\frac{1}{100}$) requires $\Omega(nk)$ time. Then, in Chapter 5, we study our generalization of the k-median problem, the online median problem, and give a constant-competitive algorithm that runs in $\Theta(n^2)$ time. In Chapter 6, we discuss our experimental study of our k-median and online median algorithm. Finally, we offer some concluding remarks in Chapter 7.

Chapter 2

Related Work

In this chapter, we give a brief overview of traditional clustering techniques as well as a survey of recent approximation algorithms for the facility location and k-median problems. Clustering is a fundamental problem in unsupervised learning that has found application in many problem domains. Approaches to clustering based on learning mixture models as well as minimizing a given objective function have both been well-studied [2, 5, 6, 8, 9, 32, 35]. In recent years, there has been significant interest in developing clustering algorithms that can be applied to the massive data sets that arise in problem domains such as bioinformatics and information retrieval on the World Wide Web. Such data sets pose an interesting challenge in that clustering algorithms must be robust as well as fast.

2.1 The k-Means Heuristic

The k-means heuristic is a widely-used clustering technique that has been used for several decades [9, 35]. Given a set of n points drawn from a Euclidean space, the k-means heuristic attempts to identify k cluster centers. The k-means heuristic works as follows. First, a set of k points is chosen as an initial solution. These points may be chosen arbitrarily, uniformly at random, or according to some other, typically simple, method. Then, this solution is iteratively improved as follows. In each iteration, the input points are partitioned into k sets by associating each point with its closest point in the current solution. The k centroids of the sets in the partition is returned as the output of the iteration. One iteration of the k-means heuristic requires $O(nk)$ time. Typically, a small number of iterations is needed until the solution converges to a local minimum.

We note that any subset of size k of the n given input points is a feasible initial solution for k-means. We make use of this fact to propose our k-median algorithm as an initialization procedure for k-means. Since the cost of the output of successive iterations of k-means is monotonically decreasing, we will see in Chapter 4 that using our k-median algorithm to initialize k-means yields a clustering algorithm that runs in $O(nk)$ time for a wide range of values of k while guaranteeing that the cost of the output is within a constant factor of optimal.

2.2 Learning Mixtures of Gaussians

A formulation of clustering that has perhaps been studied as extensively as k-means or k-median clustering is that of fitting a mixture of Gaussian distributions to a given set of points. Given a set of n points drawn from k Gaussian distributions with associated mixing weights, the problem of *learning Gaussian mixtures* asks us to compute the means, variances, and mixing weights underlying the original k Gaussians. This problem has been studied in areas such as psychology, geology, and astrophysics (see, e.g., [8, 32] for pointers to the literature). A classic approach to the problem of learning mixtures of Gaussians is to utilize a local search technique called *Expectation Maximization* (EM). The EM algorithm is often described as a "probabilistic" version of k-means [24]. The objective of the EM algorithm is to maximize the likelihood between the actual and computed parameters of the Gaussians. Just as k-means is susceptible to local minima, unless the initial parameters are chosen carefully, the EM algorithm is susceptible to local maxima. The first approximation algorithm for the problem of learning mixtures of Gaussians is due to Dasgupta [8]; his algorithm finds the means, variances, and mixing weights of the underlying Gaussians to the level of precision specified by the user.

It is interesting to note that algorithms for the k-median problem can be used for a certain model-based clustering problem as well. Arora and Kannan [2] formulate an approximation version of the problem of learning arbitrary Gaussians. Given points from a Gaussian mixture, they study the problem of identifying a set of Gaussians whose log-likelihood is within a constant factor of the log-likelihood of the original mixture. Their approach to this learning problem is to reduce it to the k-median problem and apply an existing constant-factor approximation algorithm.

2.3 Facility Location Problems

There has been much prior work on the facility location and k-median problems. The facility location and k-median problems were originally studied in the context of resource placement. It is straightforward to see that the notion of minimizing weighted average distance to cluster centers applies equally well when we view the cluster centers are resource distribution sites. In fact, the facility location and k-median problem are perhaps the two most well-known problems in the area of *discrete location theory*. The operations research community has studied problems in discrete location theory for several decades. For a survey of the area, we refer the reader to the texts by Mirchandani and Francis [38] and Love *et al.* [33]. Problems in this area have also received attention recently due to their applicability to the problem of placing resources on a network [25, 26, 27].

As mentioned in Chapter 1, we focus on the metric versions of these problems; for recent work and pointers to the literature on the general (non-metric) facility location and k-median problem, see [47]. More generally, the text by Hochbaum [18] surveys the techniques used to obtain approximation lower bounds and approximation algorithms for a number of $\mathcal{NP}$-hard optimization problems related to and including the facility location problem and k-median problem. Below we present the recent work on the metric facility location and k-median problem. The remainder of this section is organized by the algorithmic techniques used to attack these problems.

2.3.1 Linear-Programming Approaches

The first constant-factor approximation algorithm for facility location is due to Shmoys *et al.* [42] and is based on rounding the (fractional) solution to a linear program. We will refer to algorithms that take this approach as *LP-based*. Chudak [7] gives an LP-based $(1 + \frac{2}{e})$-approximation algorithm for facility location. This was the best constant factor known until the work of Charikar and Guha [5], which establishes a slightly lower approximation ratio of 1.728. The first constant-factor approximation for the k-median problem was given by Charikar *et al.* [6] and is also LP-based. That work follows a sequence of bicriteria results utilizing LP-based techniques [30, 31]. (These bicriteria results produce a configuration of size $O(k)$ with cost at most a constant factor times that of an optimal configuration of size k.) Jain and Vazirani [22] give the first nearly linear-time (in the input size) combinatorial algorithms for the facility location and k-median problems, achieving approximation ratios of 3 and 6, respectively. Their algorithm for facility location runs in $O(n^2 \log n)$ time, and their algorithm for the k-median problem runs in $O((n^2 \log n)(\ell + \log n))$, where ℓ is roughly the number of bits needed to represent any distance in the input. While the latter algorithms are combinatorial, the primal-dual approach used in their analysis is based on linear programming theory. (See [13] for an excellent introduction to the primal-dual method.)

2.3.2 Local Search and Greedy Approaches

Strategies based on local search and greedy techniques are commonly used to obtain algorithms for the facility location problem and the k-median problem. The work of Korupolu *et al.* [26] shows that a simple local search heuristic proposed by Kuehn and Hamburger [28] in the 1960s yields both a constant-factor approximation for the facility location problem and a bicriteria approximation for the k-median problem. Guha and Khuller [15] showed that greedy improvement can be used as a postprocessing step to improve the approximation guarantee of certain facility location algorithms. Subsequently, Charikar and Guha [5] achieve an approximation ratio of 1.853 for facility location by combining a local search heuristic with the best LP-based algorithm known; this algorithm runs in $\tilde{O}(n^3)$ time. Charikar and Guha also give a 4-approximation for the k-median problem by building on the techniques of Jain and Vazirani [22]; this algorithm also runs in $\tilde{O}(n^3)$ time. Subsequent to our work, Arya *et al.* [3] gave an analysis of local search that yields the first constant-factor approximation guarantee for the k-median problem. More recently, Jain *et al.* [21] gave a simple $O(n^3)$-time greedy algorithm for the facility location, obtaining the best known approximation ratio of 1.61.

2.3.3 "Sublinear-Time" Algorithms

The algorithms discussed above for the facility location and k-median problems are "superlinear" in the sense that the input size for these problems (i.e., the number of pairwise distances) is $O(n^2)$, and the algorithms discussed above all have running times of $\omega(n^2)$. In this section, we discuss "sublinear" algorithms for the facility location and k-median problem; that is, algorithms running in $o(n^2)$ time. The first constant-factor approximation algorithm for the k-median problem running in $o(n^2)$ time is due to Indyk. He gives a randomized $(O(k), O(1))$-approximate algorithm for the uniform weights k-median problem [19] that runs in $\tilde{O}(nk/\delta^2)$ time, where

δ is the desired failure probability. Indyk's algorithm combines random sampling of the input points with a black-box $(\alpha k,\ \beta)$-approximate k-median algorithm to achieve a $(2\alpha k,\ (1+\delta)(6+3\beta))$-approximate algorithm, where δ is the desired success probability. Given an $\tilde{O}(n^2)$-time black-box k-median algorithm, Indyk's algorithm runs in $\tilde{O}(nk/\delta^2)$ time. (The $\tilde{O}$-notation omits polylogarithmic factors in n and k)

Thorup [43] gives randomized constant-factor approximation algorithms for several facility location problems in a graph. In this problem setting, we are not given a metric distance function but rather a graph on the input points with m positively weighted edges from which the distances must be computed; all of Thorup's algorithms run in [43] run in $\tilde{O}(m)$ time. Thorup [43] also gives an $\tilde{O}(nk)$ time randomized constant-factor approximation algorithm for the k-median problem that we consider. As part of this k-median algorithm, Thorup gives a sampling technique that also consists of a series of sampling steps; his approach produces an $(O((k\log^2 n)/\varepsilon),\ 2+\varepsilon)$-configuration for any positive real ε with $0<\varepsilon<0.4$, but is only guaranteed to succeed with probability $\frac{1}{2}$. We note that Thorup also makes use of our facility location algorithm as the basis for a $\tilde{O}(m)$-time algorithm for the facility location problem in a graph.

Mishra *et al.* [39] show that in order to find a $(k,\ O(1))$-configuration, it is enough to take a sufficiently large sample of the input points and use it as input to a black-box $O(1)$-approximate k-median algorithm. To compute a $(k,\ O(1))$-configuration with an arbitrarily high constant probability, the required sample size is $\tilde{O}(R_d^2 k)$. In the general case, the size of the sample may be as large as n, but depending on the diameter of the input metric space, this technique can yield running times of $o(n^2)$ (e.g., if the diameter is $o(n^2/k)$).

Guha *et al.* [16] give k-median algorithms for the data stream model of computation. Under the data stream model of computation, input data is processed sequentially, and the performance of an algorithm is measured by how many passes it makes over the input and by its space requirements. For the data stream model of computation, Guha *et al.* [16] give a single-pass $O(1)$-approximate algorithm for the k-median problem that runs in $\tilde{O}(nk)$ time and requires $O(n^{\varepsilon})$ space for a positive constant ε. We also make use of a technique due to Guha *et al.* [16] that takes an $(m,\, O(1))$-configuration and extracts a $(k,\, O(1))$-configuration; they use this technique in isolation in a divide-and-conquer fashion to develop their k-median algorithms. We view the extraction technique as a postprocessing step that yields a $(k,\, O(1))$-approximate k-median algorithm given an $(m,\, O(1))$-approximate k-median algorithm. In our algorithms, we take advantage of the fact that this postprocessing step can be performed rapidly. For example, if $m = O(k)$ and the black-box algorithm requires $O(n^2)$ time, the time required for postprocessing is just $O(k^2)$.

2.4 Hardness Results

In this section, we give the known lower bounds on the approximation ratio for the facility location and k-median problem as well as lower bounds on the running time of constant-factor approximation algorithms for these problems.

Guha and Khuller [15] provide the best lower bound known of 1.463 on the approximation ratio for the facility location problem. Guha [14] gives a lower bound of $1+\frac{1}{e}$ on the approximation ratio for the k-median problem. Jain *et al.* [21] recently improved this lower bound to $1+\frac{2}{e}$.

For the facility location problem, Thorup [43] notes that it is relatively straightforward to establish an $\Omega(n^2)$ lower bound on deterministic constant-factor approximation algorithms for the facility location problem. Guha *et al.* [16] establish a lower bound of $\Omega(nk)$ for deterministic $O(1)$-approximate k-median algorithms. We note that they work with a slightly different definition of the k-median problem in which the distance between two distinct points is allowed to be 0. We adopt the view that points at distance zero are represented by a single point with commensurately higher weight; this view avoids having an infinite value for R_d. For the proof of the lower bound, Guha *et al.* [16] construct a problem instance for which optimal solution has cost 0 and reduce the problem to a graph k-partitioning problem [23]. The intuition is that any algorithm producing a k-configuration with nonzero cost is not $O(1)$-approximate. Although their lower bound arguments make use of distinct points at distance 0 (i.e., an infinite R_d), with a slight modification their proof only requires that R_d exceed n by a sufficiently large constant factor relative to the desired approximation ratio. Intuitively, with such a large setting of R_d, a deterministic k-median algorithm taking $o(nk)$ time and making just one "mistake" fails to achieve the desired approximation ratio. Our lower bounds are stronger in the sense that we focus on constructing problem instances that have small values of R_d, and then show that any randomized k-median algorithms running in $o(nk)$ time is likely to make many "mistakes" on these instances.

Chapter 3

The Facility Location Problem

In this chapter, we present a simple $O(n^2)$-time algorithm for the facility location problem that achieves an approximation ratio of 3. As stated previously, the main idea of our algorithm is to compute and use the "value" of balls about every point in the metric space. In retrospect, the idea of value is implicit in the work of Jain and Vazirani [22]. We make this idea explicit and use the values of balls to make greedy choices. Additionally, our algorithm is faster than the Jain-Vazirani algorithm by a logarithmic factor. Thorup [43] notes that it is relatively straightforward to see that $\Omega(n^2)$ time is required by any algorithm for the facility location problem that achieves a constant approximation ratio. Thus, our simple greedy algorithm is time-optimal.

The following definitions are used throughout the present chapter as well as Chapters 4 and 5.

- For any nonnegative integer m, let $[m]$ denote the set $\{i \mid 0 \leq i < m\}$.
- A ***ball*** A is a pair (x, r), where the ***center*** x of A, denoted $center(A)$, belongs to U, and the ***radius*** r of A, denoted $radius(A)$, is a nonnegative real.
- Given a ball $A = (x, r)$, we let $Points(A)$ denote the set $\{y \in U \mid d(x, y) \leq r\}$.

However, for the sake of brevity, we tend to write A instead of $\mathit{Points}(A)$. For example, we write "$x \in A$" and "$A \cup B$" instead of "$x \in \mathit{Points}(A)$" and "$\mathit{Points}(A) \cup \mathit{Points}(B)$", respectively.

- The ***value*** of a ball $A = (x, r)$, denoted $\mathit{value}(A)$, is $\sum_{y \in A}(r - d(x, y)) \cdot w(y)$.
- For any ball $A = (x, r)$ and any nonnegative real c, we define cA as the ball (x, cr).

3.1 Algorithm

In the first step of the following algorithm, we assume that there is at least one point x such that $w(x) > 0$. (The problem is trivial otherwise.) The output of the algorithm is the configuration Z_n, which we also refer to as Z. Remark: The indexing of the sets Z_i has been introduced solely to facilitate the analysis.

- For each point x in U, determine an associated ball $A_x = (x, r_x)$ such that $\mathit{value}(A_x) = f(x)$.
- Determine a bijection $\varphi : [n] \to U$ such that $r_{\varphi(i-1)} \leq r_{\varphi(i)}$, $0 < i < n$.
- Let $B_i = (x_i, r_i)$ denote the ball $A_{\varphi(i)}$, $0 \leq i < n$. Let $Z_0 = \emptyset$.
- For $i = 0$ to $n - 1$: If $Z_i \cap 2B_i = \emptyset$ then let $Z_{i+1} = Z_i \cup \{x_i\}$; otherwise, let $Z_{i+1} = Z_i$.

We now sketch a simple $O(n^2)$-time implementation of the above algorithm. For each point x, the associated radius r_x can be computed in $O(n)$ time. (This is essentially a weighted selection problem.) Thus the first step requires $O(n^2)$ time. The second step involves sorting n values and can be accomplished in $O(n \log n)$ time. The running time for the third step is negligible. Each iteration of the fourth step can be easily implemented in $O(n)$ time, for a total of $O(n^2)$ time.

3.2 Approximation Ratio

In this section we establish the following theorem.

Theorem 1 *For any configuration X, cost $(Z) \leq 3 \cdot$ cost (X).*

Proof: Immediate from Lemmas 3.2.3 and 3.2.7 below. ■

Lemma 3.2.1 *For any point x_i, there exists a point x_j in Z such that $j \leq i$ and $d(x_i, x_j) \leq 2r_i$.*

Proof: If there is no such point x_j with $j < i$, then $Z_i \cap 2B_i$ is empty, and so x_i belongs to Z. ■

Lemma 3.2.2 *Let x_i and x_j be distinct points in Z. Then $d(x_i, x_j) > 2 \cdot \max\{r_i, r_j\}$.*

Proof: Assume without loss of generality that $j < i$. Thus $r_i \geq r_j$. Furthermore, $d(x_i, x_j) > 2r_i$ since x_j belongs to Z_i and $Z_i \cap 2B_i$ is empty. ■

For any point x and any configuration X, let

$$\mathit{charge}(x, X) \quad = \quad d(x, X) + \sum_{x_i \in X} \max\{0, r_i - d(x_i, x)\}.$$

Lemma 3.2.3 *For any configuration X, $\sum_{x \in U} \mathit{charge}(x, X) \cdot w(x) =$ cost (X).*

Proof: Note that

$$\begin{aligned} \sum_{x \in U} \mathit{charge}(x, X) \cdot w(x) &= \sum_{x_i \in X} \sum_{x \in B_i} (r_i - d(x_i, x)) \cdot w(x) + \sum_{x \in U} d(x, X) \cdot w(x) \\ &= \sum_{x_i \in X} \mathit{value}(B_i) + \sum_{x \in U} d(x, X) \cdot w(x), \end{aligned}$$

which is equal to *cost* (X) since $\mathit{value}(B_i) = f(x_i)$. ■

Lemma 3.2.4 *Let x be a point, let X be a configuration, and let x_i belong to X. If $d(x, x_i) = d(x, X)$ then $\mathit{charge}(x, X) \geq \max\{r_i, d(x, x_i)\}$.*

Proof: If x does not belong to B_i, then $charge(x, X) \geq d(x, x_i) > r_i$. Otherwise, $charge(x, X) \geq d(x, x_i) + (r_i - d(x, x_i)) = r_i \geq d(x, x_i)$. ■

Lemma 3.2.5 *Let x be a point and let x_i belong to Z. If x belongs to B_i, then $charge(x, Z) \leq r_i$.*

Proof: By Lemma 3.2.2, there is no point x_j in Z such that $i \neq j$ and x belongs to B_j. The claim now follows from the definition of $charge(x, Z)$, since $d(x, Z) \leq d(x, x_i)$. ■

Lemma 3.2.6 *Let x be a point and let x_i belong to Z. If x does not belong to B_i, then $charge(x, Z) \leq d(x, x_i)$.*

Proof: The claim is immediate unless there is a point x_j in Z such that x belongs to B_j. If such a point x_j exists, then Lemmas 3.2.2 and 3.2.5 imply $d(x_i, x_j) > 2 \cdot \max\{r_i, r_j\}$ and $charge(x, Z) \leq r_j$, respectively. The claim now follows since $d(x, x_i) \geq d(x_i, x_j) - d(x, x_j) > 2r_j - r_j = r_j$. ■

Lemma 3.2.7 *For any point x and configuration X,*

$$charge(x, Z) \quad \leq \quad 3 \cdot charge(x, X).$$

Proof: Let x_i be some point in X such that $d(x, x_i) = d(x, X)$. By Lemma 3.2.1, there exists a point x_j in Z such that $j \leq i$ and $d(x_i, x_j) \leq 2r_i$.

If x belongs to B_j, then $charge(x, Z) \leq r_j$ by Lemma 3.2.5. The claim follows since $j \leq i$ implies $r_j \leq r_i$ and Lemma 3.2.4 implies $charge(x, X) \geq r_i$.

If x does not belong to B_j, then $charge(x, Z) \leq d(x, x_j)$ by Lemma 3.2.6. Thus $charge(x, Z) \leq d(x, x_i) + d(x_i, x_j) \leq d(x, x_i) + 2r_i$. The claim now follows by Lemma 3.2.4, since the ratio of $d(x, x_i) + 2r_i$ to $\max\{r_i, d(x, x_i)\}$ is at most 3. ■

Chapter 4

The k-Median Problem

Given a set of points and associated interpoint distances, let the ***median*** of the set be the point in the set that minimizes the weighted sum of distances to all other points in the set. (Remark: The median is essentially the discrete analog of the centroid, and is also called the *medoid* [36].) In this chapter, we will develop an efficient algorithm for the well-known clustering problem where the goal is to partition n weighted points into k sets such that the sum, over all points x, of the weight of x multiplied by the distance from x to the median of set containing x is minimized. This clustering problem is a variant of the classic k-median problem; the k-median problem asks us to mark k of the points such that the sum over all points x of the weight of x times the distance from x to the nearest marked point is minimized. It is straightforward to see that the optimal objective function values for the k-median problem and its clustering variant are equal, and furthermore that we can convert a solution to the k-median problem into an equal-cost solution to its clustering variant in $O(nk)$ time. We establish a lower bound of $\Omega(nk)$ time on any randomized constant-factor approximation algorithm for either the k-median problem or its clustering variant. Therefore, any constant-factor approximation algorithm for the k-median problem implies a constant-factor approximation algorithm with the same asymptotic time complexity for the clustering variant. For this reason, we focus only on the k-median problem in developing our upper bounds.

In light of the practical importance of clustering in numerous application areas, we are also motivated to ask how input characteristics such as the point weights and interpoint distances affect the complexity of the k-median problem and its clustering variant. Weighted points are useful in a number of applications; we ask the following natural question: Does allowing inputs with arbitrary point weights incur a substantial time penalty? We note that even for moderate weights, say $O(n^2)$, the naive approach of viewing a weighted point as a collection of unit-weight points increases the input size dramatically. For certain applications, the interpoint distances may lie in a relatively small range. Thus we are motivated to ask: Does constraining distances to a small range admit substantially faster algorithms? We resolve both of the above questions for a wide range of input parameters by establishing a time bound of $\Theta(nk)$ for the k-median problem and its clustering variant. Thus, in many cases having large point weights does not incur a substantial time penalty, and, we cannot hope to develop substantially faster algorithms even when the interpoint distances lie in a small range.

4.1 Summary of Results

Before summarizing the results in this chapter, we present some additional notation that we will use in this chapter as well as the next chapter. Recall that we fixed a set of points U with an associated metric distance function $d : U \times U \to \mathbb{R}$ and nonnegative function $w : U \to \mathbb{R}$. An m***-configuration*** is a configuration of size at most m. For any points x and y in U, let $w(x)$ denote the nonnegative weight of x. We denote the minimum cost of any m-configuration by OPT_m. (Recall that an m-configuration corresponds to a set of m cluster centers.) For brevity, we say that an m-configuration with cost at most $a \cdot OPT_k$ is an (m, a)***-configuration***.

We define an ***assignment*** as a function from U to U. For any assignment τ, we let $\tau(U)$ denote the set $\{\tau(x) \mid x \in U\}$. We refer to an assignment τ with $|\tau(U)| \leq m$ as a m-***assignment***. Given an assignment τ, we define the cost of τ, denoted $c(\tau)$, as $\sum_{x \in U} d(x, \tau(x)) \cdot w(x)$. It is straighforward to see that for any assignment τ, $\mathit{cost}(\tau(U)) \leq c(\tau)$. For brevity, we say that an assignment τ with $|\tau(U)| \leq m$ and cost at most $a \cdot OPT_k$ is an $(m,\ a)$-***assignment***. For an assignment τ and a set of points X, we let $c(\tau, X) = \sum_{x \in X} d(x, \tau(x)) \cdot w(x)$. A k-median algorithm is ***(m, a)-approximate*** if it produces an $(m,\ a)$-configuration. A k-median algorithm is a-***approximate*** if it is $(k,\ a)$-approximate. In light of the practical importance of clustering in the application areas mentioned previously, we also consider the given interpoint distances and point weights in our analysis. Let R_d denote the ratio of the diameter of U (i.e., the maximum distance between any pair of points in U) to the minimum distance between any pair of distinct points in U. Let R_w denote the ratio of the maximum weight of any point in U to the minimum nonzero weight of any point in U. (Remark: We can assume without loss of generality that at least one point in U has nonzero weight since the problem is trivial otherwise.) Let $r_d = 1 + \lfloor \log R_d \rfloor$ and $r_w = 1 + \lfloor \log R_w \rfloor$.

We will also say that certain claims hold *with high probability*; that is, for any positive constant ξ, we can adjust constant factors in the definition of the algorithm to achieve a failure probability less than $n^{-\xi}$.

In this chapter we present a randomized $O(1)$-approximate k-median algorithm that runs in

$$O\left(\left[n + r_d r_w \log\left(\frac{n}{k r_w}\right)\right] \max\{k, \log n\} + (k r_w)^2\right) \tag{4.1}$$

time. Note that if $k = \Omega(\log n)$, $k r_w^2 = O(n)$, and $r_d r_w \log(\frac{n}{k r_w}) = O(n)$, this time bound simplifies to $O(nk)$. Furthermore, these constraints simplify if we make the standard assumption that the interpoint distances and point weights are polynomially bounded. Then, we only need $k = \Omega(\log n)$ and $k = O(\frac{n}{\log^2 n})$ to obtain a time bound of $O(nk)$. Our algorithm succeeds *with high probability*, that is, for any positive constant ξ, we can adjust constant factors in the definition of the algorithm to achieve a failure probability less than $n^{-\xi}$.

We also establish a matching $\Omega(nk)$ lower bound on the running time of any randomized $O(1)$-approximate k-median algorithm with a nonnegligible success probability (e.g., at least $\frac{1}{100}$), subject to the requirement that R_d exceeds n/k by a sufficiently large constant factor relative to the desired approximation ratio. To obtain tight bounds for the clustering variant, we also prove an $\Omega(nk)$ time lower bound for any $O(1)$-approximate algorithm, but we only require that R_d be a sufficiently large constant relative to the desired approximation ratio. Additionally, our lower bounds assume only that $R_w = O(1)$.

The key building block underlying our k-median algorithm is a novel sampling technique that we call "successive sampling". The basic idea is to take a random sample of the points, set aside a constant fraction of the n points that are "close" to the sample, and recurse on the remaining points. We show that this technique rapidly produces a configuration whose cost is within a constant factor of optimal. Specifically, for the case of uniform weights, our successive sampling algorithm yields a $(k \log(n/k),\ O(1))$-configuration with high probability in $O(n \max\{k, \log n\})$ time.

In addition to this sampling result, our algorithms rely on an extraction technique due to Guha *et al.* [16] that uses a black box $O(1)$-approximate k-median algorithm to compute a $(k,\ O(1))$-configuration from any $(m,\ O(1))$-assignment. The black box algorithm that we use is the linear-time deterministic online median algorithm presented in Chapter 5. We note that although our online median algorithm solves a more general problem than k-median, it is still the fastest known deterministic constant-factor approximation algorithm for the k-median problem. We note that this extraction technique implies that our sampling algorithm could have significant practical benefit as a data reduction technique and thus has the possibility of obtaining a highly scalable k-median algorithm. That is, instead of considering all n points, this extraction result allows to consider just the output of our sampling algorithm, which has roughly $O(k \log n/k)$ points.

In developing our randomized algorithm for the k-median problem we first consider the special case of uniform weights, that is, where $R_w = r_w = 1$. For this special case we provide a randomized algorithm running in $O(n \max\{k, \log n\})$ time subject to the constraint $r_d \log \frac{n}{k} = O(n)$. The uniform-weights algorithm is based directly on the two building blocks discussed above: We apply the successive sampling algorithm to obtain $(k \log (n/k),\ O(1))$-configuration and then use the extraction technique to obtain a $(k,\ O(1))$-configuration. We then use this algorithm to develop a k-median algorithm for the case of arbitrary weights. Our algorithm begins by partitioning the n points into r_w power-of-2 weight classes and applying the uniform-weights algorithm within each weight class (i.e., we ignore the differences between weights belonging to the same weight class, which are less than a factor of 2 apart). The union of the r_w k-configurations thus obtained is an $(r_w k,\ O(1))$-configuration. We then make use of our extraction technique to obtain a $(k,\ O(1))$-configuration from this $(r_w k,\ O(1))$-configuration.

4.1.1 Comparison to k-Means

As mentioned in Chapter 2, approaches to clustering such as the k-means heuristic are well-studied (see, e.g., [9, 36]). The k-means heuristic is commonly used in practice due to ease of implementation, speed, and good empirical performance. Indeed, one iteration of the k-means heuristic requires just $O(nk)$ time [9]; typical implementations of the k-means heuristic make use of a small to moderate number of iterations.

However, it is easy to construct inputs with just a constant number of points that, for certain initializations of k-means, yield solutions whose cost is not within any constant factor of the optimal cost. For example, suppose we have 5 unit-weight points in $\mathbb{R}^2$ where three points are colored blue and two are colored red. Let the blue points have coordinates $(0,1)$,$(0,0)$, and $(0,-1)$, and let the red points have coordinates $(-D,0)$ and $(D,0)$. For $k = 3$, the optimal solution has cost 1, whereas the k-means heuristic, when initialized with the blue points, converges to a solution with cost $2D$ (the blue points). Since D can be arbitrarily large, in this case the k-means heuristic does not produce a solution within any constant factor of optimal. Indeed, a variety of heuristics for initializing k-means have been previously proposed, but no such initialization procedure is known to ensure convergence to a constant-factor approximate solution.

The reader may wonder whether, by not restricting the k output points to be drawn from the n input points, the k-means heuristic is able to compute a solution of substantially lower cost than would otherwise be possible. The reduction in the cost is at most a factor of two since given a k-means solution with cost C, it is straightforward to identify a set of k input points with cost at most $2C$.

For the k-means heuristic, the objective function is typically taken to be the sum of squared distances rather than distances. The reader may wonder whether this variation leads to a substantially different optimization problem. It is straightforward to show that squaring the distances of a metric space yields a distance function that is "near-metric" in the sense that all of the properties of a metric space are satisfied except that the triangle inequality only holds to within a constant factor (2, in this case). It is not difficult to show that all of our upper bounds hold, up to constant factors, for such near-metric spaces. Thus, if our algorithm is used as the initialization procedure for k-means, the cost of the resulting solution is guaranteed to be within a constant factor of optimal. Our algorithm is particularly well-suited for this purpose because its running time, being comparable to that of a single iteration of k-means, does not dominate the overall running time.

4.2 Successive Sampling

Our first result is a successive sampling algorithm that constructs an assignment that has cost $O(OPT_k)$ with high probability. We make use of this algorithm to develop our uniform weights k-median algorithm. (Remark: We assume arbitrary weights for our proofs since the arguments generalize easily to the weighted case; furthermore, the weighted result may be of independent interest.) Informally speaking, the algorithm works in sampling steps. In each step we take a small sample of the points, set aside a constant fraction the weight whose constituent points are each close to the sample, and recurse on the remaining points. Since we eliminate a constant fraction of the weight at each sampling step, the number of samples taken is logarithmic in the total weight. We are able to show that using the samples taken, it is possible to construct an assignment whose cost is within a constant factor of optimal with high probability. For the uniform weights k-median problem, our sampling algorithm runs in $O(n \max\{k, \log n\})$ time. (We give a k-median algorithm for the case of arbitrary weights in Section 4.5.)

Throughout the remainder of this paper, we use the symbols α, β, and k' to denote real numbers appearing in the definition and analysis of our successive sampling algorithm. The value of α and k' should be chosen to ensure that the failure probability of the algorithm meets the desired threshold. (See the paragraph preceding Lemma 4.3.3 for discussion of the choice of α and k'.) The asymptotic bounds established in this paper are valid for any choice of β such that $0 < \beta < 1$.

We also make use of the following definitions:

- A ***ball*** A is a pair (x, r), where the ***center*** x of A belongs to U, and the ***radius*** r of A is a nonnegative real.

- Given a ball $A = (x, r)$, we let $\mathit{Points}(A)$ denote the set $\{y \in U \mid d(x, y) \leq r\}$. However, for the sake of brevity, we tend to write A instead of $\mathit{Points}(A)$. For example, we write "$x \in A$" and "$A \cup B$" instead of "$x \in \mathit{Points}(A)$" and "$\mathit{Points}(A) \cup \mathit{Points}(B)$", respectively.

- For any set X and nonnegative real r, we define $\mathit{Balls}(X, r)$ as the set $\cup_{x \in X} A_x$ where $A_x = (x, r)$.

4.2.1 Algorithm

The following algorithm takes as input an instance of the k-median problem and produces an assignment σ such that with high probability, $c\,(\sigma) = O(\mathit{cost}\,(X))$ for any k-configuration X.

Let $U_0 = U$, and let $S_0 = \emptyset$. While $|U_i| > \alpha k'$:

- Construct a set of points S_i by sampling (with replacement) $\lfloor \alpha k' \rfloor$ times from U_i, where at each sampling step the probability of selecting a given point is proportional to its weight.

- For each point in U_i, compute the distance to the nearest point in S_i.

- Using linear-time selection on the distances computed in the previous step, compute the smallest real ν_i such that $w(\mathit{Balls}(S_i, \nu_i)) \geq \beta w(U_i)$. Let $C_i = \mathit{Balls}(S_i, \nu_i)$.

- For each x in C_i, choose a point y in S_i such that $d(x, y) \leq \nu_i$ and let $\sigma(x) = y$.

- Let $U_{i+1} = U_i \setminus C_i$.

Note that the loop terminates since $w(U_{i+1}) < w(U_i)$ for all $i \geq 0$. Let t be the total number of iterations of the loop. Let $C_t = S_t = U_t$. By the choice of C_i in each iteration and the loop termination condition, t is $O(\log(w(U)/k'))$. For the uniform demands k-median problem, t is simply $O(\log(n/k'))$. From the first step it follows that $|\sigma(U)|$ is $O(tk')$.

The first step of the algorithm can be performed in $O(nk')$ time over all iterations. In each iteration the second and third steps can be performed in time $O(|U_i|\, k')$ by using a (weighted) linear time selection algorithm. For the uniform demands k-median problem, this computation requires $O(nk')$ time over all iterations. The running times of the third and fourth steps are negligible. Thus, for the uniform demands k-median problem, the total running time of the above algorithm is $O(nk')$.

4.3 Analysis of the Successive Sampling Algorithm

The goal of this section is to establish that, with high probability, the output σ of our successive sampling algorithm has cost $O(OPT_k)$. We formalize this statement in Theorem 2 below; this result is used to analyze the algorithms of Sections 4.4 and 4.5. The proof of the theorem makes use of Lemma 4.3.3, established in Section 4.3.1, and Lemmas 4.3.5 and 4.3.11, established in Section 4.3.2.

Theorem 2 *With high probability, $c(\sigma) = O(cost(X))$ for any k-configuration X.*

Proof: The claim of Lemma 4.3.3 holds with high probability if we set $k' = \max\{k, \log n\}$ and α and β appropriately large. The theorem then follows from Lemmas 4.3.3, 4.3.5, and 4.3.11. ■

Before proceeding, we give some intuition behind the proof of Theorem 2. The proof consists of two main parts. First, Lemma 4.3.3 shows that with high probability, for i such that $0 \le i \le t$, the value ν_i computed by the algorithm in each iteration is at most twice a certain number μ_i. We define μ_i to be the minimum real for which there exists a k-configuration X contained in U_i with the property that a certain constant fraction, say $\frac{3}{4}$, of the weight of U_i is within distance μ_i

from the points of X. We note that μ_i can be used in establishing a lower bound on the cost of an optimal k-configuration for U_i. By the definition of μ_i, for any k-configuration Y, a constant fraction, say $\frac{1}{4}$, of the weight of U_i has distance at least μ_i from the points in Y. To prove Lemma 4.3.3, we consider an associated balls-in-bins problem. For each i, $1 \leq i \leq t$, we consider a k-configuration X that satisfies the definition of μ_i and for each point in X, view the points in U_i within distance μ_i as a weighted bin. Then, we view the random samples in the first step of the sampling algorithm as ball tosses into these weighted bins. We show that with $O(k)$ such ball tosses, a high constant fraction of the total weight of the bins is covered with high probability. Since the value of ν_i is determined by the random samples, it is straightforward to conclude that ν_i is within twice μ_i.

It may seem that Theorem 2 follows immediately from Lemma 4.3.3, since for each i, we can approximate μ_i within a factor of 2 with ν_i, and any optimal k-configuration can be charged a distance of at least μ_i for a constant fraction of the weight in U_i. However, this argument is not valid since for $j > i$, U_j is contained in U_i; thus an optimal k-configuration could be charged μ_i and μ_j for the same point. For the second part of the proof of Theorem 2 we provide a more careful accounting of the cost of an optimal k-configuration. Specifically, in Section 4.3.2, we exhibit t mutually disjoint sets with which we are able to establish a valid lower bound on the cost of an optimal k-configuration. That is, for each i, $1 \leq i \leq t$, we exhibit a subset of U_i that has a constant fraction of the total weight of U_i and for which an optimal k-configuration must be charged a distance of at least μ_i. Lemma 4.3.11 formalizes this statement and proves a lower bound on the cost of an optimal k-configuration, and Lemma 4.3.5 completes the proof of Theorem 2 by providing an upper bound on the cost of σ.

4.3.1 Balls and Bins Analysis

The proof of Lemma 4.3.3 below relies on bounding the failure probability of a certain family of random experiments. We begin by bounding the failure probability of a simpler family of random experiments related to the well-known coupon collector problem. For any positive integer m and any nonnegative reals a and b, let us define $f(m, a, b)$ as the probability that more than am bins remain empty after $\lceil b \rceil$ balls are thrown at random (uniformly and independently) into m bins. Techniques for analyzing the coupon collector problem (see. e.g., [40]) can be used to obtain sharp estimates on $f(m, a, b)$. However, the following simple upper bound is sufficient for our purposes.

Lemma 4.3.1 *For any positive real ε, there exists a positive real λ such that for all positive integers m and any real $b \geq m$, we have $f(m, \varepsilon, \lambda b) \leq e^{-b}$.*

Proof: Note that a crude upper bound on $f(m, \varepsilon, \lambda b)$ is given by the probability of obtaining at most $(1 - \varepsilon)m$ successes in $\lceil \lambda b \rceil$ Bernoulli trials, each of which has success probability ε. The claim then follows by choosing λ sufficiently large and applying a standard Chernoff bound. (We have in mind the following tail bound: If X is a random variable drawn from a Bernoulli distribution with n trials and each trial has success probability p, then for all δ such that $0 \leq \delta \leq 1$, $\Pr\{X \leq (1 - \delta)np\} \leq e^{-\delta^2 np/2}$; see [1, Appendix A] for a derivation.) ■

We now develop a weighted generalization of the preceding lemma. For any positive integer m, nonnegative reals a and b, and m-vector $v = (r_0, \ldots, r_{m-1})$ of nonnegative reals r_i, we define define $g(m, a, b, v)$ as follows. Consider a set of m bins numbered from 0 to $m - 1$ where bin i has associated weight r_i. Let R denote the total weight of the bins. Assume that each of $\lceil b \rceil$ balls is thrown independently at random into one of the m bins, where bin i is chosen with probability r_i/R, $0 \leq i < m$. We define $g(m, a, b, v)$ as the probability that the total weight of the empty bins after all of the balls have been thrown is more than aR.

Lemma 4.3.2 *For any positive real ε there exists a positive real λ such that for all positive integers m and any real $b \geq m$, we have $g(m, \varepsilon, \lambda b, v) \leq e^{-b}$ for all m-vectors v of nonnegative reals.*

Proof: Fix ε, b, m, and v. As in the paragraph preceding the lemma statement in Section 4.2, let $v = (r_0, \ldots, r_i)$ and let R denote the sum of the r_i's.

We will use Lemma 4.3.1 to deduce the existence of a suitable choice of λ that depends only on ε. Our strategy for reducing the claim to its unweighted counterpart will be to partition almost all of the weight associated with the m weighted bins into $\Theta(m)$ "sub-bins" of equal weight. Specifically, we let s denote $\frac{\varepsilon R}{2m}$ and for each i we partition the weight r_i associated with bin i into $\lfloor \frac{r_i}{s} \rfloor$ *complete* sub-bins of weight s and one *incomplete* sub-bin of weight less than s. Furthermore, when a ball is thrown into a particular bin, we imagine that the throw is further refined to a particular sub-bin of that bin, where the probability that a particular sub-bin is chosen is proportional to its weight.

Note that the total weight of the incomplete sub-bins is less than $\varepsilon R/2$. Furthermore, we can assume without loss of generality that $\varepsilon \leq 1$, since the claim holds vacuously for $\varepsilon > 1$. It follows that less than half of the total weight R lies in incomplete sub-bins. Thus, by a standard Chernoff bound argument, for any positive real λ' we can choose λ sufficiently large to ensure that the following claim holds with probability of failure at most $e^{-b}/2$ (i.e., half the desired failure threshold appearing in the statement of the lemma): At least $\lambda' b$ of the $\lceil \lambda b \rceil$ balls are thrown into complete sub-bins.

Let m' denote the number of complete sub-bins. Since at least half of the total weight R belongs to complete sub-bins, we have $m/\varepsilon \leq m' \leq 2m/\varepsilon$. Accordingly, by a suitable application of Lemma 4.3.1, we can establish the existence of a positive real λ' (depending only on ε) such that, after at least $\lambda' b$ balls have landed in complete sub-bins, the probability that the number of empty complete sub-bins exceeds $\varepsilon m'/2$ is at most $e^{-b}/2$.

From the claims of the two preceding paragraphs, we can conclude that there exists a λ (depending only on ε) such that the following statement holds with probability of failure at most e^{-b}: The number of empty complete sub-bins is at most $\varepsilon m'/2$. Note that the total weight of the complete sub-bins is at most $s \cdot \frac{\varepsilon}{2} \cdot \frac{2t}{\varepsilon} = \varepsilon R/2$. As argued earlier, the total weight of the incomplete sub-bins is also at most $\varepsilon R/2$. Thus, there exists a positive real λ such that after $\lceil \lambda b \rceil$ ball tosses, the probability that the total weight of the empty bins is more than εR is at most e^{-b}. ■

For the remainder of this section, we fix a positive real γ such that $\beta < \gamma < 1$. For $0 \leq i \leq t$, let μ_i denote a nonnegative real such that there exists a k-configuration X for which the following properties hold: (1) the total weight of all points x in U_i such that $d(x, X) \leq \mu_i$ is at least $\gamma w(U_i)$; (2) the total weight of all points x in U_i such that $d(x, X) \geq \mu_i$ is at least $(1-\gamma)w(U_i)$. (Note that such a μ_i is guaranteed to exist.) Lemma 4.3.3 below establishes the main probabilistic claim used in our analysis of the algorithm of Section 4.2.1. We note that the lemma holds with high probability by taking $k' = \max\{k, \lceil \log n \rceil\}$ and α and β appropriately large.

Lemma 4.3.3 *For any positive real ξ, there exists a sufficiently large choice of α such that $\nu_i \leq 2\mu_i$ for all i, $0 \leq i \leq t$, with probability of failure at most $e^{-\xi k'}$.*

Proof: Fix i and let X denote a k-configuration such that $w(\mathit{Balls}(X, \mu_i)) \geq \gamma w(U_i)$. Let us define each point y in U_i to be *good* if it belongs to $\mathit{Balls}(X, \mu_i)$, and *bad* otherwise. Let G denote the set of good points. We associate each good point y with its closest point in X, breaking ties arbitrarily. For each point x in X, let A_x denote the set of good points associated with x; note that the sets A_x form a partition of G. Recall that S_i denotes the ith set of sample points chosen by the algorithm. For any x in X, we say that S_i *covers* A_x iff $S_i \cap A_x$ is nonempty. For any point y, we say that S_i *covers* y iff there exists an x in X such that y belongs to A_x and S_i covers A_x. Let G' denote the set of points covered by S_i; note that $G' \subseteq G$.

We will establish the lemma by proving the following claim: For any positive reals ε and ξ, there exists a sufficiently large choice of α such that $w(G') \geq (1 - \varepsilon)w(G)$ with probability of failure at most $e^{-\xi k'}$. This claim then implies the lemma because β (the factor appearing in the definition of ν_i) is less than γ (the factor appearing in the definition of μ_i) and for all points y covered by S_i, $d(y, S_i) \leq 2\mu_i$.

It remains to prove the preceding claim. First, note that the definition of μ_i implies that at least a γ fraction of the total weight is associated with good points. Thus, a standard Chernoff bound argument implies that for any positive reals λ and ξ, there exists a sufficiently large choice of α such that at least $\lambda k'$ of the $\lfloor \alpha k' \rfloor$ samples associated with the construction of S_i are good with probability of failure at most $e^{-\xi k'}/2$.

To ensure that $w(G')$ is at least $(1-\varepsilon)w(G)$ with failure probability $e^{-\xi k'}/2$, we can apply Lemma 4.3.2 by viewing each sample associated with a good point in S_i as a ball toss and each set A_x as a bin with weight $w(A_x)$. The claim then follows. ∎

4.3.2 Upper and Lower Bounds on Cost

In this section we provide an upper bound on the cost of the assignment σ as well a lower bound on the cost of an optimal k-configuration. Lemmas 4.3.4 and 4.3.5 establish the upper bound on $c(\sigma)$, while the rest of the section is dedicated to establishing the lower bound on the cost of an optimal k-configuration.

Lemma 4.3.4 *For all i such that $0 \leq i \leq t$, $c(\sigma, C_i) \leq \nu_i w(C_i)$.*

Proof: Observe that

$$\begin{aligned} c(\sigma, C_i) &= \sum_{x \in C_i} d(x, \sigma(x)) \cdot w(x) \\ &\leq \sum_{x \in C_i} \nu_i \cdot w(x) \\ &= \nu_i w(C_i), \end{aligned}$$

where the second step follows from the definition of C_i and the construction of $\sigma(x)$.

■

Lemma 4.3.5

$$c(\sigma) \leq \sum_{0 \leq i \leq t} \nu_i w(C_i)$$

Proof: Observe that $c(\sigma) = \sum_{0 \leq i \leq t} c(\sigma, C_i) \leq \sum_{0 \leq i \leq t} \nu_i w(C_i)$. The first step follows since the sets C_i, $0 \leq i \leq t$, form a partition of U. The second step follows from Lemma 4.3.4. ■

We now focus on establishing a lower bound on the cost of an optimal k-configuration. Throughout the remainder of this section we fix an arbitrary k-configuration X. For all i such that $0 \leq i \leq t$, we let F_i denote the set $\{x \in U_i \mid d(x, X) \geq \mu_i\}$, and for any integer $m > 0$, we let F_i^m denote $F_i \setminus (\cup_{j>0} F_{i+jm})$ and we let $G_{i,m}$ denote the set of all integers j such that $0 \leq j \leq t$ and j is congruent to i modulo m.

Lemma 4.3.6 *Let i, j, ℓ, and m be integers such that $0 \leq \ell \leq t$, $m > 0$, $i \neq j$, and i and j belong to $G_{\ell,m}$. Then $F_i^m \cap F_j^m = \emptyset$.*

Proof: Without loss of generality, assume that $i < j$. Then, by definition, $F_i^m = F_i \setminus (\cup_{s>0} F_{i+sm})$. Since $F_j^m \subseteq F_j$ and $j = i + sm$ for some positive integer s, it follows that F_i^m and F_j^m do not intersect. ■

Lemma 4.3.7 *Let i be an integer such that $0 \leq i \leq t$ and let Y be a subset of F_i. Then $w(F_i) \geq (1-\gamma)w(U_i)$ and $\mathit{cost}\,(X,Y) \geq \mu_i w(Y)$.*

Proof: First, note that by the definition of μ_i, $w(F_i)$ is at least $(1-\gamma)w(U_i)$. By the definition of F_i, $d(y,X) \geq \mu_i$ for any y in F_i. Thus $\mathit{cost}\,(X,Y) = \sum_{y \in Y} d(y,X) \cdot w(y) \geq \mu_i w(Y)$. ■

Lemma 4.3.8 *For all integers ℓ and m such that $0 \leq \ell \leq t$ and $m > 0$,*

$$\mathit{cost}\Big(X, \cup_{i \in G_{\ell,m}} F_i^m\Big) \quad \geq \quad \sum_{i \in G_{\ell,m}} \mu_i w(F_i^m).$$

Proof: By Lemma 4.3.6, for all ℓ and m such that $0 \leq \ell \leq t$ and $m > 0$,

$$\mathit{cost}\Big(X, \cup_{i \in G_{\ell,m}} F_i^m\Big) \quad = \quad \sum_{i \in G_{\ell,m}} \mathit{cost}\,(X, F_i^m)\,.$$

By Lemma 4.3.7, $\mathit{cost}\,(X, F_i^m) \geq \mu_i w(F_i^m)$, and the claim follows. ■

For the remainder of the section, let $r = \lceil \log_{(1-\beta)}((1-\gamma)/3) \rceil$.

Lemma 4.3.9 *For all i such that $0 \leq i \leq t$, $w(F_{i+r}) \leq \frac{1}{3} w(F_i)$.*

Proof: Note that $w(F_{i+r}) \leq w(U_{i+r}) \leq (1-\beta)^r w(U_i) \leq \frac{(1-\beta)^r}{1-\gamma} w(F_i)$, where the last step follows from Lemma 4.3.7. The claim then follows by the definition of r. ■

Lemma 4.3.10 *For all i such that $0 \leq i \leq t$, $w(F_i^r) \geq \frac{w(F_i)}{2}$.*

Proof: Observe that

$$\begin{aligned} w(F_i^r) &= w(F_i \setminus \cup_{j>0} F_{i+jr}) \\ &\geq w(F_i) - \sum_{j>0} \frac{w(F_i)}{3^j} \\ &\geq \frac{w(F_i)}{2}, \end{aligned}$$

where the second step follows from Lemma 4.3.9. ■

Lemma 4.3.11 *For any k-configuration X,*

$$cost(X) \geq \frac{1-\gamma}{2r} \sum_{0 \leq i \leq t} \mu_i w(C_i).$$

Proof: Let $\ell = \arg\max_{0 \leq \ell < r} \{\sum_{i \in G_{\ell,r}} w(F_i^r)\}$ and fix a k-configuration X. Then $cost(X)$ is at least

$$\begin{aligned}
cost\left(X, \cup_{i \in G_{\ell,r}} F_i^r\right) &\geq \sum_{i \in G_{\ell,r}} \mu_i w(F_i^r) \\
&\geq \frac{1}{r} \sum_{0 \leq i \leq t} \mu_i w(F_i^r) \\
&\geq \frac{1}{2r} \sum_{0 \leq i \leq t} \mu_i w(F_i) \\
&\geq \frac{1-\gamma}{2r} \sum_{0 \leq i \leq t} \mu_i w(U_i) \\
&\geq \frac{1-\gamma}{2r} \sum_{0 \leq i \leq t} \mu_i w(C_i),
\end{aligned}$$

where the first step follows from Lemma 4.3.8, the second step follows from averaging and the choice of ℓ, the third step follows from Lemma 4.3.10, the fourth step follows from Lemma 4.3.7, and the last step follows since $C_i \subseteq U$. ■

4.4 An Efficient Algorithm for the Case of Uniform Weights

In this section we use the sampling algorithm of Section 4.2, a black-box k-median algorithm and algorithm Modified-Small-Space of Appendix A to obtain a fast k-median algorithm for the case of uniform weights. We note that algorithm Modified-Small-Space and the accompanying analysis is a slight generalization of results obtained by Guha *et al.* [16]. Informally speaking, algorithm Modified-Small-Space works in two phases. First, we use an (a, $O(1)$)-approximate k-median algorithm on the input to compute ℓ (a, $O(1)$)-configurations. Then, we construct a new k-median problem instance from these (a, $O(1)$)-configurations and use an $O(1)$-approximate k-median algorithm to compute a k-configuration. We are able to show that this k-configuration is actually a (k, $O(1)$)-configuration.

We obtain our uniform weights k-median algorithm by applying our sampling algorithm in Step 2 of algorithm Modified-Small-Space and the online median algorithm of Chapter 5 in Step 4. (Remark: Although the online median problem is a generalization of the k-median problem, the $O(n^2)$-time online median algorithm of Chapter 5 is still the fastest deterministic algorithm for the k-median problem.) We set the parameter ℓ of algorithm Modified-Small-Space to 1 and parameter k' of our sampling algorithm to $\max\{k, \log n\}$. By Theorem 2, the output of our sampling algorithm is an (m, $O(1)$)-assignment with high probability, where $m = O(\max\{k, \log n\} \log (n/k))$. Thus, by Theorem 6, the resulting k-median algorithm is $O(1)$-approximate with high probability.

We now analyze the running time of the above algorithm on inputs with uniform weights. The time required to compute the output assignment σ in Step 2 is $O(n \max\{k, \log n\})$. We note that the weight function required in Step 3 of Modified-Small-Space can be computed during the execution of the sampling algorithm without increasing its running time. The deterministic online median algorithm of Chapter 5 requires $O(|\sigma(U)|^2 + |\sigma(U)|\, r_d)$ time. The total time taken by the algorithm is therefore

$$\begin{aligned} & O(nk' + |\sigma(U)|^2 + |\sigma(U)|\, r_d) \\ = \quad & O(nk' + k'^2 \log^2 (n/k) + r_d k' \log (n/k)) \\ = \quad & O(nk' + r_d k' \log (n/k)), \end{aligned}$$

where the first step follows from the analysis of our sampling algorithm for the case of uniform weights. By the choice of k', the overall running time is $O((n + r_d \log (n/k)) \max\{k, \log n\})$. Note that if $k = \Omega(\log n)$ and $r_d \log (n/k) = O(n)$, this time bound simplifies to $O(nk)$.

4.5 An Efficient Algorithm for the Case of Arbitrary Weights

The algorithm developed in Sections 4.2 and 4.4 is $O(1)$-approximate for the k-median problem with arbitrary weights. However, the time bound established for the case of uniform weights does not apply to the case of arbitrary weights because the running time of the successive sampling procedure is slightly higher in the latter case. (More precisely, the running time of the sampling algorithm of Section 4.2 is $O(nk' \log \frac{w(U)}{k'})$ for the case of arbitrary weights.) In this section, we use the uniform-weight algorithm developed in Sections 4.2 and 4.4 to develop a k-median algorithm for the case of arbitrary weights that is time optimal for a certain range of k. We first give an informal description of the algorithm, which consists of three main steps. First, we partition the input points according to weight into r_w sets. Next, we run our uniform weights k-median algorithm on each of the resulting sets, and show that the union of the resulting outputs is an $(O(kr_w), O(1))$-configuration. We then obtain a $(k, O(1))$-configuration by creating a problem instance from the $(O(kr_w), O(1))$-configuration computed in the previous step and then feeding this problem instance as input to an $O(1)$-approximate k-median algorithm.

We now give a precise description of our k-median algorithm. Let $\mathcal{A}$ be the uniform weights k-median algorithm of Sections 4.2 and 4.4, and let $\mathcal{B}$ be an $O(1)$-approximate k-median algorithm.

- Compute sets B_i for $0 \leq i < r_w$ such that for all $x \in B_i$, $2^i \leq w(x) \leq 2^{i+1}$.

- For $i = 0, 1 \ldots r_w - 1$: Run $\mathcal{A}$ with B_i as the set of input points, d as the distance function, 2^{i+1} as the fixed weight, and the parameter $k' = \max\{k, \lceil \log n \rceil\}$; let Z_i denote the output. Let ϕ_i denote the assignment induced by Z_i, that is, $\phi_i(x) = y$ iff y is in Z_i and $d(x, Z_i) = d(x, y)$. For a point x, if $x \in Z_i$, let $w_{\phi_i}(x) = w(\phi_i^{-1}(x))$, otherwise let $w_{\phi_i}(x) = 0$.

- Let ϕ be the assignment corresponding to the union of the assignments ϕ_i defined in the previous step, and let w_ϕ denote the weight function corresponding to the union of the weight functions w_{ϕ_i}. Run $\mathcal{B}$ with $\phi(U)$ as the set of input points, d as the distance function, and w_ϕ as the weight function. Output the resulting k-configuration.

Note that in the second step, k' is defined in terms of n (i.e., $|U|$) and not $|B_i|$. Thus, the argument of the proof of Theorem 2 implies that $\mathcal{A}$ succeeds with high probability in terms of n. Assuming that r_w is polynomially bounded in n, with high probability we have that every invocation of $\mathcal{A}$ is successful.

We now observe that the above algorithm corresponds to algorithm Modified-Small-Space with the parameter ℓ is set to r_w, the uniform weights algorithm of Section 4.4 is used in step 2 of Small-Space, and the online median algorithm presented in Chapter 5 is used in step 4 of Small-Space. Thus,Theorem 6 implies that the output of $\mathcal{B}$ is a (k, $O(1)$)-configuration with high probability.

We now discuss the running time of the above algorithm. It is straightforward to compute the sets B_i in $O(n)$ time. Our uniform weights k-median algorithm requires $O((|B_i| + r_d \log \frac{|B_i|}{k})k')$ time to compute Z_i, so the time required for all invocations of $\mathcal{A}$ is

$$\begin{aligned} & O\left(\sum_{0\le i<r_w} (|B_i| + r_d \log(|B_i|/k))\,k'\right) \\ = \quad & O\left(r_w\left(\frac{nk'}{r_w} + r_d k' \log\left(\frac{n}{kr_w}\right)\right)\right) \\ = \quad & O\left(\left(n + r_d r_w \log\frac{n}{kr_w}\right)k'\right). \end{aligned}$$

(The first step follows from the fact that the sum is maximized when $|B_i| = n/r_w$.) Note that each weight function w_{ϕ_i} can be computed in $O(|B_i|\,k)$ time; it follows that w_ϕ can be computed in $O(nk)$ time. We employ the online median algorithm of Chapter 5 as the black-box k-median algorithm $\mathcal{B}$. Since $|\phi(U)|$ is at most kr_w, the time required for the invocation of $\mathcal{B}$ is $O((kr_w)^2 + kr_w r_d)$. It follows that the overall running time of the algorithm is as stated in Equation (4.1).

4.6 Lower Bounds

In this section, we give lower bounds for the k-median problem and its clustering variant. Throughout the section, we refer to the clustering variant as the *k-clustering problem*. Recall that the k-clustering problem asks us to partition the input points such that the sum, over all sets in the partition, of the weight of a point times the distance to the median of its set, is minimized. Since any k-median solution can be converted into a solution for the k-clustering problem in $O(nk)$ time, in developing our upper bounds it was sufficient to consider only the k-median problem. Unfortunately this reduction is not useful for the present purpose of establishing $\Omega(nk)$ lower bounds; accordingly, in this section we consider the problems separately.

For both the k-clustering problem and the k-median problem, we establish a lower bound of $\Omega(nk)$ time on any randomized algorithm that is $O(1)$-approximate with even a negligible probability. Since the overall objective of this paper is to study the complexity of approximate clustering in terms of the four parameters n, k, R_d, and R_w, it is desirable for the metric spaces associated with our lower bound arguments to have small values for both R_d and R_w. In terms of R_w, we achieve this goal completely, since all of the input distributions that we consider below have uniform weights, that is, $R_w = 1$. For the k-clustering problem, our lower bounds are established with R_d equal to a constant (sufficiently large relative to the desired approximation ratio); this is clearly best possible up to a constant factor. For the k-median problem, our lower bound requires R_d to exceed n/k by a sufficiently large constant factor relative to the desired approximation ratio.

In our proofs, we assume an oracle model of computation in which the algorithm is charged only for asking the oracle the distance between a pair of points.

We refer to each call to the oracle as a *probe*. By a generalization of Yao's technique [46] due to Mackenzie [34], we can establish a lower bound of p on the success probability of a randomized algorithm by exhibiting an input distribution for which every deterministic algorithm has a success probability of at most p. (The intuition underlying this reduction is that the success probability of a randomized algorithm is just a convex combination of the success probabilities of a number of deterministic algorithms.) Thus in what follows, we restrict our attention to exhibiting "hard" distributions for determinstic algorithms. All of the problems considered in this section take the same input as the k-median problem. Our lower bounds also hold for the non-uniform case since for each choice of n and k, we exhibit a probability distribution over the set of n-point metric spaces on which no deterministic algorithm making a sufficiently small number of probes can achieve more than a negligible probability of success.

For any positive real $\ell > 1$, it is convenient to define a metric space to be *ℓ-simple* if the following conditions hold: (1) all of the points have unit weight; (2) the points of the metric space can be partitioned into equivalence classes such that the distance between any pair of distinct points is 1 if the points belong to the same equivalence class, and ℓ otherwise. Thus, any ℓ-simple metric space has $R_d = \ell$ and $R_w = 1$. Our lower bounds are all based on ℓ-simple input distributions for some appropriately chosen value of ℓ.

In order to establish a lower bound for the k-clustering problem, we find it convenient to introduce a problem that we call the *k-matching problem*. The input to the k-matching problem is the same as the input to the k-clustering problem. The output is a partition of the n input points into a collection of disjoint pairs and singletons, subject to the constraint that there are at most k singletons. We refer to such an output as a *k-matching*. The *cost* of a k-matching is defined as the sum, over all output pairs of points (x, y), of $d(x, y) \cdot \min\{w(x), w(y)\}$. The goal of the

k-matching problem is to compute a minimum-cost k-matching.

Given an algorithm for the k-clustering problem, consider the associated k-matching algorithm defined as follows: (1) run the k-clustering algorithm to partition the n input points into at most k clusters; (2) arbitrarily partition each even-sized cluster into a number of pairs; (3) arbitrarily partition each odd-sized cluster into a singleton and a number of pairs; (4) return the k-matching formed by the singletons and pairs computed in the previous two steps. Using the triangle inequality, it is straightforward to prove that the cost of the k-matching produced by this algorithm is at most the cost of the k-clustering computed in step (1) (i.e., the sum over all points x of the weight of x multiplied by the distance from x to the medoid of its cluster). Furthermore, this k-matching algorithm uses exactly the same number of probes as the associated k-clustering algorithm. Below we will exhibit an input distribution with respect to which any deterministic k-matching algorithm making a sufficiently small number of probes has only a negligible probability of computing a k-matching with cost within a constant factor of the cost of the optimal clustering. By the foregoing reduction from the k-matching problem to the k-clustering problem, such a result implies that any deterministic k-clustering algorithm running on the same input distribution and making the same small number of probes has only the same negligible probability of computing a k-clustering with cost within a constant factor of optimal.

In order to state and prove our lower bounds it is convenient to introduce a shorthand notation for expressing certain kinds of statements. In particular, for any statement S, we define an associated statement, which we refer to as the *P-claim* S, as follows: For all positive reals ε and c, there exist positive reals δ and γ and positive integers n_0 and a such that for all positive integers n and k for which $n \geq n_0$ and $1 < k < n$, there exists a probability distribution D over the set of ℓ-simple n-point metric spaces where $\ell = \gamma$ such that any deterministic k-matching algorithm

$\mathcal{A}$ making at most δnk probes on an input drawn uniformly at random from D, the statement S holds with probability at least $1-\varepsilon$. (We remark that a given P-claim S need not contain the parameter c. We also remark that if the P-claims S and T hold, then the P-claim $S \wedge T$ holds.)

We define a P'-claim in the same way as a P-claim except that the restriction on k is strengthened to $1 < k < \frac{n}{2}$. Similarly, a P''-claim is a variant of a P-claim in which the restriction on k is $\frac{n}{2} \le k < n$. Note that for any statement S, the P'-claim S and the P''-claim S imply the P-claim S.

Finally, for addressing the k-median problem we define Q-, Q'-, and Q''-claims in an analogous manner, where the algorithm $\mathcal{A}$ is assumed to be a k-median algorithm rather than a k-matching algorithm, and ℓ is defined to be $\frac{\gamma n}{k}$ instead of γ.

The rest of this section is devoted to proving the following two theorems.

Theorem 3 *The P-claim "the cost of the k-matching solution computed by $\mathcal{A}$ is more than c times the cost of an optimal k-clustering solution" holds.*

Theorem 4 *The Q-claim "the cost of the k-median solution computed by $\mathcal{A}$ is more than c times the cost of an optimal k-median solution" holds.*

The proof of the first theorem follows from Lemmas 4.6.1 and 4.6.2 below. The proof of the second theorem follows from Lemmas 4.6.3 and 4.6.4.

Lemma 4.6.1 *The P'-claim "the cost of the k-matching solution computed by $\mathcal{A}$ is more than c times the cost of an optimal k-clustering solution" holds.*

Proof Sketch: Let D denote the distribution of ℓ-simple n-point metric spaces where each point is independently placed into one of k equivalence classes uniformly at random. Given an input instance drawn from D, the cost of an optimal k-clustering solution is easily seen to be $n-k$.

Let us define a point x to be *clean* with respect to an execution of algorithm $\mathcal{A}$ if the following two conditions are satisfied: (1) there is no point y such that $d(x, y) = 1$ and $\mathcal{A}$ has probed $d(x, y)$; (2) $\mathcal{A}$ has probed the distance between x and at most εk other points.

It is not difficult to establish the following P'-claim: "At least $(1-\varepsilon)n$ points are clean". Since $\mathcal{A}$ is a k-matching algorithm it outputs at least $n - k \geq n/2$ pairs. This observation, together with the preceding P'-claim, implies the P'-claim "At least $n/3$ of the pairs produced by $\mathcal{A}$ consist of two clean points." Note that each such output pair of clean points independently contributes a cost of ℓ to the cost of the k-matching produced by $\mathcal{A}$ with probability at least $1 - \frac{1}{k(1-\varepsilon)}$, since a clean point is equally likely to belong to any of the at least $k(1 - \varepsilon)$ equivalence classes (those for which $\mathcal{A}$ has not probed a distance between the given clean point and some point in the equivalence class). The claim of the lemma now follows by choosing constants appropriately (i.e., by setting δ, γ, and n_0 to appropriate functions of ε and c) and applying a standard Chernoff bound argument. ■

Lemma 4.6.2 *The P''-claim "the cost of the k-matching solution computed by $\mathcal{A}$ is more than c times the cost of an optimal k-clustering solution" holds.*

Proof Sketch: The proof of the preceding lemma does not readily extend to large values of k, so we employ a somewhat different approach. In this case we define the input distribution D by randomly partitioning the n points into k clusters (i.e., equivalence classes), $n-k$ of which are pairs, and $2k-n$ of which are singletons. As in the proof of Lemma 4.6.1, the cost of an optimal k-clustering solution is $n - k$.

Let us assume for the sake of simplicity that n is a multiple of $2a$. (Remark: It is not difficult to modify our argument to handle general n.) For the sake of the analysis, it is useful to think of sampling from the input distribution D via the following three-stage process: (1) randomly partition the n points into $\frac{n}{2a}$ *supergroups* of size $2a$; (2) randomly partition each supergroup into a pairs; (3) pick a random set of $k - \frac{n}{2}$ pairs and split them to obtain $2k - n$ singletons. In what follows we refer to these pairs and singletons as *input-pairs* and *input-singletons*, in order to avoid confusion with the pairs and singletons computed by algorithm $\mathcal{A}$, which we refer to as *output-pairs* and *output-singletons*.

We define a supergroup to be *interesting* if it contains at least one input-pair. Note that there are at least $\frac{n-k}{a}$ interesting supergroups. Let us define a supergroup to be *red* if it contains at least one output-pair; otherwise, it is *blue*.

If there are i blue supergroups then at least i output-pairs either span distinct supergroups or contain at least one input-singleton; it follows that the cost of the k-matching produced by $\mathcal{A}$ is at least $i\ell$. If at least half (say) of the interesting supergroups are blue, this argument is sufficient to establish the lemma. Thus, in what follows, we may assume that at least half of the interesting supergroups are red.

Let us define a supergroup to be *clean* with respect to an execution of algorithm $\mathcal{A}$ if $\mathcal{A}$ does not probe the distance between any two points in the supergroup. It is not difficult to establish the following P''-claim: "At least a $1 - \varepsilon$ fraction of the interesting supergroups are clean." By this P''-claim and the assumption of the previous paragraph, we establish the P''-claim "at least one-third of the interesting supergroups are clean and red".

Let G denote a clean interesting red supergroup and let (x, y) denote an output-pair that belongs to G (such a pair exists since G is red). If x is an input-singleton then the cost of pair (x, y) is ℓ, and we can attribute this cost to G. Otherwise, x belongs to some input-pair (x, z), and algorithm $\mathcal{A}$ pays ℓ for the pair (x, y) unless $y = z$. But the probability that $y = z$ is $\frac{1}{2a-1}$ since G is clean. Furthermore, the event that $y = z$ is independent of the analogous events defined for other clean interesting red supergroups. Thus each clean interesting red supergroup independently contributes, with probability at least $1 - \frac{1}{2a-1}$, a cost of at least ℓ to the total cost of the k-matching produced by $\mathcal{A}$. The claim of the lemma now follows by choosing constants appropriately and applying a standard Chernoff bound argument. ∎

Lemma 4.6.3 *The Q'-claim "the cost of the k-median solution computed by $\mathcal{A}$ is more than c times the cost of an optimal k-median solution" holds.*

Proof Sketch: Let D denote the distribution of ℓ-simple n-point metric spaces associated with the following partitioning scheme: (1) independently place each of the n points into one of $\lfloor k/2 \rfloor$ *tentative equivalence classes* uniformly at random; (2) randomly select $\lceil k/2 \rceil$ *special* points and move each of these special points into a singleton equivalence class. Note that for any such instance, the cost of an optimal k-median solution is $n - k$.

We define a point x to be *clean* with respect to an execution of algorithm $\mathcal{A}$ if there is no point y belonging to the same tentative equivalence class as x for which $\mathcal{A}$ has probed $d(x, y)$.

It is not difficult to establish the following pair of Q'-claims: (1) at least $(1-\varepsilon)n$ points are clean; (2) at least $(1-\varepsilon)\lceil k/2 \rceil$ of the special points are clean.

Let X denote the random variable corresponding to the set of clean points, and let Y denote the remaining points. Let Z denote the random variable corresponding to the set of special clean points. We now argue that the conditional distribution of Z given X and $|Z|$ has a simple structure, namely, Z is a uniformly random subset of X of size $|Z|$. This claim holds because the definition of a clean point implies that the behavior of algorithm $\mathcal{A}$ is the same no matter which size-$|Z|$ subset of X is equal to Z. Combining this claim with the results of the preceding paragraph, it is straightforward to establish the Q'-claim "$\mathcal{A}$ fails to output $\frac{1}{4}$ (say) of the clean special points."

Note that each special point that does not appear in the output of $\mathcal{A}$ contributes ℓ to the cost of the k-median solution computed by $\mathcal{A}$. Thus we obtain the Q'-claim "the cost of the solution computed by $\mathcal{A}$ is at least $(1-\varepsilon)k\ell/8$". Choosing γ sufficiently large (depending on c), the claim of the lemma then follows since $\ell = \gamma n/k$. ■

Lemma 4.6.4 *The Q''-claim "the cost of the k-median solution computed by $\mathcal{A}$ is more than c times optimal" holds.*

Proof Sketch: This proof is similar to that of Lemma 4.6.2 above. We define the input distribution D in the same manner, as well as the following terms: supergroup, clean supergroup, interesting supergroup, input-pair, input-singleton. As before, note that at least $\frac{n-k}{a}$ of the supergroups are interesting.

We define the *input-weight* of a supergroup as the number of input-pairs and input-singletons that it contains. We define the *output-weight* of a supergroup as the size of its intersection with the k-median solution computed by $\mathcal{A}$. We define the *discrepancy* of a supergroup as its input-weight minus its output-weight. Note that the sum of the discrepancies of all supergroups is zero since the total input-weight and the total output-weight are both equal to k. A supergroup is *balanced* if it has discrepancy 0.

If the total discrepancy of the supergroups with positive discrepancy is s then it is straightforward to prove that the cost of the k-median solution computed by $\mathcal{A}$ is at least $s\ell$. If s is at least one-quarter of the number of interesting supergroups then this argument is sufficient to establish the claim of the lemma. Thus in what follows we may assume that s is less than one-quarter of the number of interesting supergroups. Under this assumption, at least half of the interesting supergroups are balanced (since at most one-quarter of them can have negative discrepancy).

It is not difficult to establish the following Q''-claim: "At least a $1-\varepsilon$ fraction of the interesting supergroups are clean." Combining this with the conclusion of the preceding paragraph we obtain the Q''-claim “at least one-third of the interesting supergroups are clean and balanced”.

Let G denote a clean interesting balanced supergroup with i input-pairs and j input-singletons. Thus the input-weight and output-weight of G is $i + j$ (since G is balanced), and $i > 0$ (since G is interesting). In order to avoid paying a cost of ℓ for servicing any of the points in supergroup G, the subset of G of size $i + j$ contained in the output of $\mathcal{A}$ has to include exactly one point out of each of the i input-pairs, and all of the j input-singletons. Since G is clean, the probability that $\mathcal{A}$ produces such an output is 2^i divided by $\binom{2a}{i}$. Given the constraints on i, namely, $1 \leq i \leq a$, this probability is at most $1/a$. Furthermore, the event that $\mathcal{A}$ produces such an output is independent the analogous events defined for other clean interesting balanced supergroups. Thus each clean interesting balanced supergroup independently contributes, with probability at least $1/a$, a cost of at least ℓ to the total cost of the k-median solution produced by $\mathcal{A}$. The claim of the lemma now follows by choosing constants appropriately and applying a standard Chernoff bound argument. ■

Chapter 5

The Online Median Problem

In this chapter, we formulate and study the online median problem, our generalization of the k-median problem. To give some intuition behind the online median problem, we first present it as a problem of resource location and show how it generalizes the k-median problem. Recall that the k-median problem was originally studied in the context of facility placement [38]. Given a set of n cities and the intercity distances, the k-median problem addressed the problem of how to open k facilities for a certain resource while minimizing the average distance from any city to its closest facility. To see how the online median problem generalizes the k-median problem in the context of facility placement, consider the following scenario. Suppose we wish to open a new chain of stores in a city with n neighborhoods, and that we have a good estimate of the demand for our product in each neighborhood. In determining where to locate the stores, our high-level strategy is to minimize the the demand-weighted average distance from a customer to the nearest store. Our business plan is to start with one store, and then to gradually add new stores as allowed by our profits. (Remark: We will never move a previously established store.) Thus our configuration of stores may change over time, and hence the ratio between the service cost of our configuration and that of an optimal same-size configuration may also change. The goal of the online median problem is to choose a site for each new store so that the maximum value of this ratio is minimized.

In the context of clustering, the k-median problem asks us to identify k cluster centers so that the average cluster diameters is minimized. In many settings, it is desirable to browse a given data set at differing levels of granularity (i.e., number of clusters). Additionally, it may be the case that the ideal value of k is not known *a priori*. A naive approach to this problem would be to simply run a k-median algorithm for all values of k such that $1 \leq k < n$. However, by the lower bound of Chapter 4, such an approach has a running time of $\Omega(n^3)$. We show that it is possible to do significantly better. In the context of clustering, the online median problem asks us to compute an ordering of the points such that if we take a prefix of length k of the ordering as a set of k cluster centers, the ratio between the cost of these cluster centers and the optimal i cluster centers is minimized. We give an algorithm for the online median problem that is constant-competitive and runs in $O(n^2)$ time. Thus, in $O(n^2)$ our algorithm allows us, for all values of i, to obtain a set of i cluster centers whose cost is within a constant factor of optimal. We note that any constant-competitive algorithm for the online median problem requires $\Omega(n^2)$ time. (The proof of this fact follows by applying the lower bound of Chapter 4 for the case $k = \frac{n}{2}$.) Thus, our online median algorithm is time-optimal.

Recall that we have fixed a set of points U with an associated distance function $d : U \times U \to \mathbb{R}$ and nonnegative function $w : U \to \mathbb{R}$. For the online median problem, it will prove to be useful to consider a slightly more general class of distance functions in which the triangle inequality is relaxed to the following "λ-approximate" triangle inequality, where $\lambda \geq 1$: For any sequence of points $\langle x_0, \ldots, x_m \rangle$, $d(x_0, x_m) \leq \lambda \cdot \sum_{0 \leq i < m} d(x_i, x_{i+1})$. We refer to such a distance function as a ***λ-approximate metric***. We will assume that d is a λ-approximate metric.

5.1 A Hierarchically Greedy Strategy

In Chapter 3, we found that a simple greedy algorithm yields interesting results for the facility location problem. The most obvious greedy algorithm for the online median problem is to select as the next point in the ordering the one that minimizes the objective function. Unfortunately, this algorithm gives an unbounded competitive (resp., approximation) ratio for the online median (resp., k-median) problem. To see this, consider an instance consisting of $n > 3$ points, one "red" and the rest "blue", such that the following conditions are satisfied: the red point has weight 0; each blue point has weight 1; the distance from the red point to any blue point is 1, and the distance between any pair of distinct blue points is 2. The aforementioned greedy algorithm chooses the red point first in the ordering, since that gives a cost of $n-1$ while choosing any other point gives a cost of $2n-4$. But then the ratio for a configuration of size $n-1$ is unbounded since the greedy cost is 1 and the optimal cost is 0. (This example also shows that no online median algorithm can achieve a competitive ratio below $2 - \frac{2}{n-1}$.)

We show that a more careful choice of the point, which we call *hierarchically greedy*, works well. Let Δ (resp., δ) denote the largest (resp., smallest) distance between two distinct points in the metric space. We define a certain ball about each point, and select a ball A of maximum value. But rather than simply choosing the center of ball A as the next point in the ordering, we apply the approach recursively to select a point within a region defined by A. At each successive level of recursion, we consider geometrically smaller balls about the remaining candidate points. Within $O(\log \frac{\Delta}{\delta})$ levels of recursion, we arrive at a ball containing a single point, and we return this point as the next one in the ordering. Note that whereas the greedy algorithm discussed in the previous paragraph makes a single greedy choice to select a point, the hierarchically greedy algorithm makes $O(\log \frac{\Delta}{\delta})$ greedy choices per point.

Throughout this chapter, let λ, α, β, and γ denote real numbers satisfying the following inequalities.

$$\lambda \geq 1 \tag{5.1}$$

$$\alpha > 1+\lambda \tag{5.2}$$

$$\beta \geq \frac{\lambda(\alpha-1)}{\alpha-1-\lambda} \tag{5.3}$$

$$\gamma \geq \left(\frac{\alpha^2\beta+\alpha\beta}{\alpha-1}+\alpha\right)\lambda \tag{5.4}$$

The online median algorithm of Section 5.2 below makes use of the following additional definitions.

- A ***child*** of a ball (x, r) is any ball $(y, \frac{r}{\alpha})$ where $d(x, y) \leq \beta r$.
- For any point x and any configuration X, let *isolated*(x, X) denote the ball $(x, \frac{d(x,X)}{\gamma})$. We let *isolated*$(x, \emptyset)$ denote the ball $(x, \max_{y \in U} d(x, y))$.
- For any nonempty sequence ϱ, we let *head*(ϱ) (resp., *tail*(ϱ)) denote the first (resp., last) element of ϱ.

5.2 Algorithm

Let $Z_0 = \emptyset$. For $i = 0$ to $n-1$, execute the following steps:

- Let σ_i denote the singleton sequence $\langle A \rangle$ where A is a maximum value ball in $\{\mathit{isolated}(x, Z_i) \mid x \in U \setminus Z_i\}$.
- While the ball *tail*(σ_i) has more than one child, append a maximum value child of *tail*(σ_i) to σ_i.
- Let $Z_{i+1} = Z_i \cup \{\mathit{center}(\mathit{tail}(\sigma_i))\}$.

The output of the online median algorithm is a collection of point sets Z_i such that $|Z_i| = i$, $0 \leq i \leq n$, and $Z_i \subseteq Z_{i+1}$, $0 \leq i < n$. Note that it is sufficient for an implementation of the algorithm to maintain the ball $\mathit{tail}(\sigma_i)$, as opposed to the entire sequence σ_i. The sequence σ_i has been introduced in order to facilitate the analysis.

We discuss two implementations of the online median algorithm in Section 5.5. The first implementation has a running time that is slightly superlinear in the input size. The second implementation has a running time that is linear in the input size, but assumes a (linear) preprocessing phase in which all distances are rounded down to the nearest integral power of λ. (Note that for the preprocessing phase to be well-defined, we require $\lambda > 1$.) If the input distance function is a metric, it is straightforward to see that such rounding produces a λ-approximate metric.

5.3 Competitive Ratio

Before proceeding with the analysis, we introduce a number of additional definitions.

- Let z_i denote the unique point in $Z_{i+1} \setminus Z_i$, $0 \le i < n$.
- For any configuration X and set of points Y, let $\mathit{cost}\,(X, Y) = \sum_{y \in Y} d(y, X) \cdot w(y)$.
- For any configuration X, we partition U into $|X|$ sets $\{\mathit{cell}(x, X) \mid x \in X\}$ as follows: For each point y in U, we choose a point x in X such that $d(y, X) = d(y, x)$ and add y to $\mathit{cell}(x, X)$.
- For any configuration X, point x in X, and set of points Y, we let $\mathit{in}(x, X, Y)$ denote $\mathit{cell}(x, X) \cap \mathit{isolated}(x, Y)$ and $\mathit{out}(x, X, Y)$ as $\mathit{cell}(x, X) \setminus \mathit{in}(x, X, Y)$.
- For any configuration X and set of points Y, we let $\mathit{in}(X, Y)$ denote the set $\cup_{x \in X} \mathit{in}(x, X, Y)$ and we let $\mathit{out}(X, Y)$ denote the set $U \setminus \mathit{in}(X, Y)$.

We now give some intuition underlying the definitions of the sets $\mathit{cell}(x, X)$, $\mathit{in}(x, X, Y)$, and $\mathit{out}(x, X, Y)$. The sets $\mathit{cell}(x, X)$ partition the input points; for each point x in X, $\mathit{cell}(x, X)$ contains the points with x as a closest neighbor in X (with ties broken arbitrarily). The sets $\mathit{in}(x, X, Y)$ and $\mathit{out}(x, X, Y)$ partition the set $\mathit{cell}(x, X)$ into two disjoint sets. In our arguments, we will consider the sets $\mathit{in}(x, X, Y)$ and $\mathit{out}(x, X, Y)$ with X as an arbitrary configuration, and Y as $Z_{|X|}$. Then the set $\mathit{out}(x, X, Z_{|X|})$ is defined as the complement of $\mathit{in}(x, X, Z_{|X|})$ in $\mathit{cell}(x, X)$). Thus, we can interpret the points in $\mathit{out}(x, X, Z_{|X|})$ to be the points that are "near" $Z_{|X|}$. For any point y in $\mathit{out}(X, Z_{|X|})$, it is relatively straightforward (see Lemma 5.3.1) to show that $d(y, Z_{|X|})$ (i.e., the distance to the configuration $Z_{|X|}$ computed by our online median algorithm) is within a constant factor of $d(y, X)$. We devote considerably more effort to show that the cost incurred by $Z_{|X|}$ for the points in $\mathit{in}(x, X, Z_{|X|})$ is within a constant factor of optimal. Recall that $\mathit{in}(x, X, Z_{|X|})$ corresponds to the points in $\mathit{cell}(x, X)$ that are contained in the ball $\mathit{isolated}(x, Z_{|X|})$. Suppose that $\mathit{isolated}(x, Z_{|X|})$ has radius r. Then, by the definition of $\mathit{isolated}(x, Z_{|X|})$, the points contained in $\mathit{in}(x, X, Z_{|X|})$ are exactly the points in $\mathit{cell}(x, X)$ that are in the ball (x, r) but have distance strictly greater than γr to any point in $Z_{|X|}$. Thus, the points in $\mathit{in}(x, X, Z_{|X|})$ are those points in $\mathit{cell}(x, X)$ that are "far" from $Z_{|X|}$. Accounting for the cost incurred by $Z_{|X|}$ for the points $\mathit{in}(X, Z_{|X|})$ will comprise the majority of the proofs in this section and the following section.

We now present our main result, Theorem 5 below. In order to minimize the competitive ratio of $2\lambda(\gamma + 1)$ implied by the theorem, we set λ to 1, set α to $2 + \sqrt{3}$ and set β and γ to the right-hand sides of Equations (5.3) and (5.4), respectively. We thereby establish a competitive ratio of below 29.86 for the online median problem. In Section 5.5 we describe an implementation of the online median algorithm for which the parameter λ is required to be strictly greater than 1. The degradation in the competitive ratio that results by setting λ greater than 1 can be made arbitrarily small by choosing λ sufficiently close to 1.

Theorem 5 *For any configuration X, $\mathit{cost}\left(Z_{|X|}\right) \leq 2\lambda(\gamma+1) \cdot \mathit{cost}\,(X)$.*

Proof: Let $Y = \mathit{in}(X, Z_{|X|})$ and let $Y' = \mathit{out}(X, Z_{|X|}) = U \setminus Y$. Then, by definition, we have $\mathit{cost}\,(X) = \mathit{cost}\,(X, Y) + \mathit{cost}\,(X, Y')$ and that $\mathit{cost}\left(Z_{|X|}\right) = \mathit{cost}\left(Z_{|X|}, Y\right) + \mathit{cost}\left(Z_{|X|}, Y'\right)$. Thus the theorem follows immediately from Lemmas 5.3.2, 5.3.4, and 5.3.5 below. ■

Lemma 5.3.1 *For any configuration X, point x in X, and point y in $\mathit{out}(x, X, Z_{|X|})$,*

$$d(y, Z_{|X|}) \quad \leq \quad \lambda(\gamma+1) \cdot d(y, X).$$

Proof: Let $\mathit{isolated}(x, Z_{|X|}) = (x, r)$. Note that $d(x, y) > r$. Also, by the definition of $\mathit{isolated}(x, Z_{|X|})$, there is a point z in $Z_{|X|}$ such that $d(x, z) = \gamma r$. Hence $d(y, z) \leq \lambda[d(x, y) + d(x, z)] = \lambda[d(x, y) + \gamma r] < \lambda[d(x, y) + \gamma \cdot d(x, y)] = \lambda(\gamma+1) \cdot d(x, y) = \lambda(\gamma+1) \cdot d(y, X)$, where the last step follows since y is in $\mathit{cell}(x, X)$. The claim follows since $d(y, z) \geq d(y, Z_{|X|})$. ■

Lemma 5.3.2 *For any configuration X,*

$$\mathit{cost}\left(Z_{|X|}, \mathit{out}(X, Z_{|X|})\right) \quad \leq \quad \lambda(\gamma+1) \cdot \mathit{cost}\left(X, \mathit{out}(X, Z_{|X|})\right).$$

Proof: Summing the inequality of Lemma 5.3.1 over all y in $\mathit{out}(x, X, Z_{|X|})$, we obtain

$$\mathit{cost}\left(Z_{|X|}, \mathit{out}(x, X, Z_{|X|})\right) \quad \leq \quad \lambda(\gamma+1) \cdot \mathit{cost}\left(X, \mathit{out}(x, X, Z_{|X|})\right).$$

The claim now follows by summing the above inequality over all x in X. ■

Lemma 5.3.3 *For any configuration X and point x in X, we have that $cost\left(Z_{|X|}, in(x, X, Z_{|X|})\right)$ is at most*

$$\lambda(\gamma+1)[cost\left(X, in(x, X, Z_{|X|})\right) + value(isolated(x, Z_{|X|}))].$$

Proof: Assume that $isolated(x, Z_{|X|}) = (x, r)$. Note that $d(x, y) = \gamma r$ for some y in $Z_{|X|}$. Thus, for any z in $isolated(x, Z_{|X|})$, $d(y, z) \le \lambda[d(y, x) + d(x, z)] \le \lambda(\gamma+1)r$, where the last step follows from our bound on $d(x, y)$ and the definition of $isolated(x, Z_{|X|})$. It follows that $cost\left(Z_{|X|}, in(x, X, Z_{|X|})\right)$ is at most $\lambda(\gamma+1)$ times

$$\begin{aligned}\sum_{z \in in(x,X,Z_{|X|})} r \cdot w(z) \quad &\le \sum_{z \in in(x,X,Z_{|X|})} d(x,z) \cdot w(z) + \\ &\quad \sum_{z \in isolated(x,Z_{|X|})} (r - d(x,z)) \cdot w(z) \\ &= cost\left(X, in(x, X, Z_{|X|})\right) + value(isolated(x, Z_{|X|})).\end{aligned}$$

■

Lemma 5.3.4 *For any configuration X and point x in X, we have that $cost\left(Z_{|X|}, in(X, Z_{|X|})\right)$ is at most*

$$\lambda(\gamma+1)[cost\left(X, in(X, Z_{|X|})\right) + \sum_{x \in X} value(isolated(x, Z_{|X|}))].$$

Proof: The claim follows by summing the inequality of Lemma 5.3.3 over all x in X. ■

Our main technical lemma is stated below. The proof is given in the next section.

Lemma 5.3.5 *For any configuration X, $\sum_{x \in X} value(isolated(x, Z_{|X|})) \le cost\,(X)$.*

5.4 Proof of Lemma 5.3.5

In this section we establish our main technical lemma, Lemma 5.3.5. Before proceeding we give some intuition behind Lemma 5.3.5 and the supporting lemmas. Informally, Lemma 5.3.5 yields an upper bound on the value of certain balls that contain points that are "far" from $Z_{|X|}$, where X is an arbitrary configuration. The upper bound we obtain states that the value associated with these points is at most $\mathit{cost}\,(X)$. Thus, in combination with Lemmas 5.3.2 and 5.3.4 we can conclude that $\mathit{cost}\left(Z_{|X|}\right)$ is $O(\mathit{cost}\,(X))$. To prove Lemma 5.3.5 we argue that for each ball that contains points "far" from $Z_{|X|}$ (i.e., $\mathit{isolated}(x, Z_{|X|})$ for each x in X), it is possible to identify a ball with commensurately high value that is "far" from X. More precisely, we construct a matching between the points in $Z_{|X|}$ and X and show that for each point x in $X \setminus Z_{|X|}$ (i.e., points in X which were considered but not chosen by our online median algorithm), we can identify a ball A_x appearing in some sequence $\sigma_i < |X|$ such that (i) $\mathit{value}(A_x) \geq \mathit{isolated}(x, Z_{|X|})$, (ii) $\mathit{cost}\,(X, A_x) \geq \mathit{value}(A_x)$, and (iii) all such balls A_x are disjoint. Intuitively, we will identify these balls by making use of the greedy manner in which our online median algorithm chooses the balls in the sequences σ_i, $0 \leq i < |X|$.

Lemma 5.4.1 *Let $A = (x, r)$ belong to σ_i. Then $d(x, Z_i) \geq \gamma r$.*

Proof: Let z be a point in Z_i such that $d(x, z) = d(x, Z_i)$. If $A = \mathit{head}(\sigma_i)$ then $A = \mathit{isolated}(x, Z_i)$ and the result is immediate. Otherwise, let $B = (y, s)$ denote the predecessor of A in σ_i and assume inductively that $d(y, Z_i) \geq \gamma s$. Note that $d(x, y) \leq \beta s$ and $s = \alpha r$. Thus $d(x, Z_i) = d(x, z) \geq d(y, z)/\lambda - d(x, y) \geq (\gamma/\lambda - \beta)\alpha r \geq \gamma r$, where the last step follows from Equation (5.4). ■

Lemma 5.4.2 *Let $A = (x, r)$ belong to σ_i and let $B = (y, s)$ belong to σ_j. If $i < j$ and $d(x, y) \leq r + s$, then the following claims hold: (i) $\mathit{radius}(\mathit{head}(\sigma_j)) \leq \frac{r}{\alpha}$;*

(ii) $A \neq tail(\sigma_i)$; (iii) the successor of A in σ_i, call it C, satisfies $value(C) \geq value(head(\sigma_j))$.

Proof: Let $head(\sigma_j) = (y', s')$. For part (i), we begin by deriving upper and lower bounds on $d(y', z_i)$. For a lower bound on $d(y', z_i)$, note that $d(y', z_i) \geq d(y', Z_j)$ (since $i < j$) and $d(y', Z_j) \geq \gamma s'$ by Lemma 5.4.1. To derive an upper bound on $d(y', z_i)$, we first let P denote the prefix of sequence σ_j ending with ball B, and let S denote the suffix of sequence σ_i beginning with ball A. We then apply the λ-approximate triangle inequality to the sequence of points $\langle y', \ldots, y, x, \ldots, z_i \rangle$ where the prefix $\langle y', \ldots, y \rangle$ corresponds to the centers of the balls in P, and the suffix $\langle x, \ldots, z_i \rangle$ corresponds to the centers of the balls in S. By repeated application of the definition of a child, and using the given upper bound on $d(x, y)$, we obtain

$$\begin{aligned} d(y', z_i) &\leq \lambda \left[\beta \left(s' + \frac{s'}{\alpha} + \cdots + \alpha s\right) + s + r + \beta \left(r + \frac{r}{\alpha} + \cdots\right)\right] \\ &\leq \left[\frac{\alpha\beta}{\alpha - 1} \cdot (r + s') + r\right] \lambda. \end{aligned}$$

Combining the bounds on $d(y', z_i)$ and applying Equation (5.4), we obtain

$$\left(\frac{\alpha^2\beta + \alpha\beta}{\alpha - 1} + \alpha\right) \lambda s' \quad \leq \quad \left[\frac{\alpha\beta}{\alpha - 1} \cdot (r + s') + r\right] \lambda.$$

Multiplying through by $(\alpha - 1)/\lambda$ and rearranging, we get $r \geq \frac{\alpha^2\beta + \alpha^2 - \alpha}{\alpha\beta + \alpha - 1} \cdot s' = \alpha s'$, establishing the claim.

For part (ii), note that $d(x, y) \leq r + \frac{r}{\alpha} < \beta r$ by part (i) and Equation (5.3). Thus A has at least two children; the claim follows.

For part (iii), we obtain an upper bound on $d(x, y')$ by applying the λ-approximate triangle inequality to the sequence of points $\langle y', \ldots, y, x \rangle$, where the prefix $\langle y', \ldots, y \rangle$ corresponds to the centers of the balls in P (as defined in part (i) above). By repeated application of the definition of a child and by the given upper bound on $d(x, y)$, we observe that

$$d(x, y') \quad \leq \quad \lambda \left[r + s + \left(\alpha s + \alpha^2 s + \cdots + s'\right) \beta\right].$$

Then, by using Equations (5.2) and (5.3) and part (i), we observe that

$$\begin{aligned}\lambda\left[r+s+\left(\alpha s+\alpha^2 s+\cdots+s'\right)\beta\right] &\le \lambda r+\frac{\alpha\beta\lambda}{\alpha-1}\cdot s'\\ &\le \lambda r+\frac{\alpha\beta\lambda}{\alpha-1}\cdot\frac{r}{\alpha}\\ &\le \left(\frac{\beta}{\alpha-1}+1\right)\lambda r.\end{aligned}$$

Observe that $\left(\frac{\beta}{\alpha-1}+1\right)\lambda r$ is at most βr by Equation (5.3). It then follows that $\mathit{head}(\sigma_j)$ is contained in a child of A. Thus $\mathit{value}(C) \ge \mathit{value}(\mathit{head}(\sigma_j))$. ∎

For ease of notation, throughout the remainder of this section we fix a configuration X, and let k denote $|X|$. We now describe a ***pruning procedure*** that we use for the purpose of analyzing our online median algorithm. The pruning procedure takes as input the k sequences σ_i, $0 \le i < k$, and produces as output k sequences τ_i, $0 \le i < k$. The sequence τ_i is initialized to σ_i, $0 \le i < k$. The (nondeterministic) pruning procedure then performs a number of iterations. In a general iteration, the pruning procedure checks whether there exist two balls $A = (x, r)$ and $B = (y, s)$ in distinct sequences τ_i and τ_j, respectively, such that $i < j$ and $d(x, y) \le r + s$. If not, the pruning procedure terminates. If so, the sequence τ_i is redefined as the proper suffix of (the current) τ_i beginning at the successor of A. Note that part (ii) of Lemma 5.4.2 ensures that the pruning procedure is well-defined. Furthermore, the procedure is guaranteed to terminate since each iteration reduces the length of some sequence τ_i.

Lemma 5.4.3 *Let $A = (x, r)$ belong to τ_i and let $B = (y, s)$ belong to τ_j. If $i < j$ then $d(x, y) > r + s$.*

Proof: Immediate from the definition of the pruning procedure. ∎

Lemma 5.4.4 *Each sequence τ_i is nonempty.*

Proof: Immediate from part (ii) of Lemma 5.4.2 and the definition of the pruning procedure. ■

Lemma 5.4.5 *Let x be a point and assume that $0 \le i < j \le n$. Then*

$$\mathit{value}(\mathit{isolated}(x, Z_i)) \quad \ge \quad \mathit{value}(\mathit{isolated}(x, Z_j)).$$

Proof: Since Z_i is contained in Z_j, we have that $\mathit{radius}(\mathit{isolated}(x, Z_i))$ is at least $\mathit{radius}(\mathit{isolated}(x, Z_j))$. The claim follows. ■

Lemma 5.4.6 *Let x be a point and assume that $0 \le i < k$. Then*

$$\mathit{value}(\mathit{head}(\sigma_i)) \quad \ge \quad \mathit{value}(\mathit{isolated}(x, Z_k)).$$

Proof: First, suppose that x belongs to Z_i. In this case there is nothing to prove since $\mathit{radius}(\mathit{isolated}(x, Z_i)) = 0$, and thus $\mathit{value}(\mathit{isolated}(x, Z_i)) = 0$. If x is not in Z_i, then $\mathit{value}(\mathit{head}(\sigma_i)) \ge \mathit{value}(\mathit{isolated}(x, Z_i))$ by the definition of the online median algorithm, and the claim follows by Lemma 5.4.5. ■

Lemma 5.4.7 *Let x be a point and assume that $0 \le i < k$. Then*

$$\mathit{value}(\mathit{head}(\tau_i)) \ge \mathit{value}(\mathit{isolated}(x, Z_k)).$$

Proof: We prove that the claim holds before and after each iteration of the pruning procedure. Initially, $\tau_i = \sigma_i$ and the claim holds by Lemma 5.4.6. If the claim holds before an iteration of the pruning procedure, then it holds after the iteration by part (iii) of Lemma 5.4.2. ■

A ball $A = (x, r)$ is defined to be ***covered*** iff $d(x, X) < r$. A ball is ***uncovered*** iff it is not covered.

Lemma 5.4.8 *For any uncovered ball $A = (x, r)$, $\mathit{cost}\,(X, A) \ge \mathit{value}(A)$.*

Proof: Note that $cost\,(X, A) \geq \sum_{y \in A} d(y, X) \cdot w(y) \geq \sum_{y \in A}(r - d(y, x)) \cdot w(y) = value(A)$. ■

Let I denote the set of all indices i in $[k]$ such that some ball in τ_i is covered. We now construct a matching between the sets $[k]$ and X as follows. First, for each i in I, we match i with a point x in X that belongs to the last covered ball in the sequence τ_i. (Note that such a point x is guaranteed to exist by the definition of I. Furthermore, Lemma 5.4.3 ensures that we do not match the same point with more than one index.) Second, for each i in $[k] \setminus I$ in turn, we match i with an arbitrary unmatched point x in X.

We now construct a function ϱ mapping each point x in X to an uncovered ball. For each x in X that is matched with an index i in $[k] \setminus I$, we set $\varrho(x)$ to $head(\tau_i)$. For each x in X that is matched with an index i in I, we set $\varrho(x)$ to the successor of the last covered ball in τ_i unless $tail(\tau_i)$ is covered, in which case we set $\varrho(x)$ to the ball $(x, 0)$.

Lemma 5.4.9 *For any pair of distinct points x and y in X, $\varrho(x) \cap \varrho(y) = \emptyset$.*

Proof: Immediate from Lemma 5.4.3 and the fact that the ball $(x, 0)$ is contained in $tail(\tau_i)$. ■

Lemma 5.4.10 *For any point x in X, $value(\varrho(x)) \geq value(isolated(x, Z_k))$.*

Proof: If x is matched with an index i in $[k] \setminus I$, the claim follows by Lemma 5.4.7. If x is matched with an index i in I, we consider two cases. If $tail(\tau_i)$ is covered, then $x = z_i$ since $tail(\tau_i)$ has exactly one child. The claim follows since $\varrho(x) = isolated(x, Z_k) = (x, 0)$. If $tail(\tau_i)$ is uncovered, then the predecessor of $\varrho(x)$ in τ_i, call it $A = (y, r)$, exists and contains x. It follows that $value(\varrho(x)) \geq value(B)$, where $B = (x, r/\alpha)$ is the child of A centered at x. Let $C = (x, s)$ denote the ball $isolated(x, Z_k)$. Below we complete the proof of the claim by showing that $r/\alpha \geq s$, which implies that $B \supseteq C$ and hence $value(B) \geq value(C)$.

It remains to prove that $r/\alpha \geq s$ in the final case considered above. We prove the claim by deriving upper and lower bounds on $d(x, z_i)$. Let S be the suffix of the sequence τ_i beginning with the ball A. For the upper bound, we apply the triangle inequality to the sequence of points $\langle x, y, \ldots, z_i \rangle$, where the suffix $\langle y, \ldots, z_i \rangle$ consists of the centers of the balls in S. We then obtain that

$$\begin{aligned} d(x, z_i) &\leq \lambda\left(r + \beta\left(r + \frac{r}{\alpha} + \cdots\right)\right) \\ &\leq \left(1 + \frac{\alpha\beta}{\alpha - 1}\right)\lambda r, \end{aligned}$$

which is less than $\gamma r/\alpha$ by Equation (5.4). The desired inequality follows since $d(x, z_i) \geq \gamma s$ by the definition of C. ■

Lemmas 5.4.8, 5.4.9, and 5.4.10 together yield a proof of Lemma 5.3.5.

5.5 Time Complexity

In this section we describe two implementations of the online median algorithm given in Section 5.2. Throughout this section, let ℓ denote the quantity $\log \frac{\Delta}{\delta}$. The first implementation runs in $O((n + \ell) \cdot n \log n)$ time. The second implementation runs in $O(n^2 + \ell n)$ time and assumes an $O(n^2)$-time preprocessing phase in which all distances are rounded down to the nearest integral power of λ. To analyze the running time of the implementations given below, we make use of the following lemma.

Lemma 5.5.1 *Let $A = (x, r)$ be a child of a ball B in sequence σ_i and let $A' = (x, r')$ be a child of a ball B' in sequence σ_j. If $i < j$ then $r \geq (\alpha + 1 + \frac{1}{\beta})r'$.*

Proof: We first obtain an upper bound on $d(x, z_i)$ by applying the λ-approximate triangle inequality to a sequence of points consisting of the centers of the balls in the suffix of σ_i beginning with ball A. Thus $d(x, z_i) \leq \lambda\beta\,(r + r/\alpha + \cdots) \leq$

$\lambda\alpha\beta r/(\alpha-1)$. By Lemma 5.4.1 and since $j > i$, we get that $\gamma r' \leq d(x, Z_j) \leq d(x, z_i)$. Combining these inequalities and using Equation (5.4), we obtain

$$\begin{aligned} r &\geq \frac{(\alpha-1)\gamma}{\lambda\alpha\beta} \cdot r' \\ &\geq \frac{\alpha-1}{\alpha\beta} \cdot \left(\frac{\alpha^2\beta + \alpha\beta}{\alpha - 1} + \alpha \right) \lambda \cdot r' \\ &= (\alpha + 1 + \frac{1}{\beta}) r'. \end{aligned}$$

■

In the first implementation, for each point x in U, we sort the remaining points by their distance from x. The total sorting time is $O(n^2 \log n)$. Using these sorted arrays, we can compute the value of any given ball in $O(\log n)$ time. We also maintain the distance from x to the nearest point in Z_i. Note that $d(x, Z_{i+1})$ can be determined in constant time given $d(x, Z_i)$ and z_i. The total time to maintain such distances is thus $O(n^2)$. It follows that the first step of each iteration can be implemented in $O(n)$ time. The total time for the second step is $O(\log n)$ times the sum over all balls A appearing in some sequence σ_i, $0 \leq i < n$, of the number of children of A. By Lemma 5.5.1, it is straightforward to see that the latter sum is $O(\ell n)$, and thus the total time for the second step is $O(\ell n \log n)$. The running time of the third step is negligible. Thus the running time of the first implementation is $O((n + \ell) \cdot n \log n)$, as claimed above.

For the second implementation, note that after the preprocessing phase, there are $O(\ell)$ distinct distances. Thus, for each point x, $O(n + \ell)$ time is sufficient to construct an $O(\ell)$-sized table that can be used to compute the value of any ball (x, r) in $O(1)$ time. It follows that the total time for the second step can be improved to $O(\ell n)$. The running time of the second implementation is therefore $O(n^2 + \ell n)$, which is linear in the size of the input (in bits).

5.6 Weakly λ-Approximate Metrics

The analysis in this chapter assumes that the (nonnegative, symmetric) distance function d approximately satisfies the triangle inequality. Recall that we defined a "λ-approximate" triangle inequality for $\lambda \geq 1$ as follows: For any sequence of points $x_0, \ldots, x_m$ in U, $d(x_0, x_m) \leq \lambda \cdot \sum_{0 \leq i < m} d(x_i, x_{i+1})$. We refer to such a distance function as a ***λ-approximate metric***.

In this section, we show that the results in this chapters as well as Chapters 3 and 4 hold to within constant factors for an even weaker form of the triangle inequality. We say that a distance function d satisfies a "weakly λ-approximate" triangle inequality if for any x, y, and z, $d(x, z) \leq \lambda(d(x, y) + d(y, z))$. We will say that such a (nonnegative, symmetric) distance function is a ***weakly λ-approximate metric***. Such distance functions are of use in extending our results to other objective functions. For example, the well-known k-means heuristic [9] has a sum of squared distances in its objective function. It is straightforward to show that squaring the distances in a metric yields a weakly 2-approximate metric. Thus, the results in this section show that our analysis also holds, to within constant factors, with respect to the k-means objective function. (Remark: More generally, it is not hard to show that raising the distances in a metric to any constant power yields a weakly $O(1)$-approximate metric.)

Lemmas 5.6.1 and 5.6.2 below establish that the approximation results in this paper hold, up to constant factors, even for weakly λ-approximate metrics. Recall that in Chapter 3, we make use of the triangle inequality and the λ-approximate triangle inequality on sequences of points to derive upper bounds on the distances between pairs of points. In most cases, we consider constant-length sequences of points to derive our upper bounds. In such cases Lemma 5.6.1 below shows that a weakly λ-approximate metric is sufficient to guarantee that our upper bounds hold to within constant factors. Unfortunately, Lemma 5.6.1 alone is not sufficient to generalize our upper bounds based on nonconstant-length sequences of points, which arise in Lemmas 5.4.2, 5.4.10, and 5.5.1. For these cases we require Lemma 5.6.2 below. Lemmas 5.6.1 and 5.6.2 together show that the upper bounds derived in Lemmas 5.4.2, 5.4.10, and 5.5.1 still hold up to constant factors given only a weakly λ-approximate triangle inequality.

Lemma 5.6.1 *Let d be a weakly λ-approximate metric, and let $x_0, x_1, \ldots, x_m$ be points with $m \geq 1$. Then, $d(x_0, x_m) \leq \lambda^{\lceil \log_2 m \rceil} \cdot \sum_{0 \leq i < m} d(x_i, x_{i+1})$.*

Proof: We will prove the lemma by induction. The base case, $m = 1$, is trivial. For the induction step, assume that for any sequence of points $y_0, \ldots, y_i$, $1 \leq i < m$, $d(y_0, y_i) \leq \lambda^{\lceil \log_2 i \rceil} \sum_{0 \leq j < i} d(y_j, y_{j+1})$. Then,

$$\begin{aligned}
d(x_0, x_m) &\leq \lambda \left(d(x_0, x_{\lceil \frac{m}{2} \rceil}) + d(x_{\lceil \frac{m}{2} \rceil}, x_m) \right) \\
&\leq \lambda \left(\lambda^{\lceil \log_2 \lceil \frac{m}{2} \rceil \rceil} \left(\sum_{0 \leq j < \lceil \frac{m}{2} \rceil} d(x_j, x_{j+1}) \right) + \right. \\
&\qquad \left. \lambda^{\lceil \log_2 \lfloor \frac{m}{2} \rfloor \rceil} \left(\sum_{\lceil \frac{m}{2} \rceil \leq j < m} d(x_j, x_{j+1}) \right) \right) \\
&\leq \lambda \cdot \lambda^{\lceil \log_2 m \rceil - 1} \sum_{0 \leq j < m} d(x_j, x_{j+1}) \\
&= \lambda^{\lceil \log_2 m \rceil} \sum_{0 \leq j < m} d(x_j, x_{j+1}).
\end{aligned}$$

The first step follows from the weakly λ-approximate triangle inequality. The second step follows by applying the induction hypothesis twice (note that $m \geq 2$ implies that $0 < \lceil \frac{m}{2} \rceil < m$, so the induction hypothesis is applicable). The last step follows from the fact that $\lceil \log_2 \lceil \frac{m}{2} \rceil \rceil = \lceil \log_2 m \rceil - 1$. ■

If λ and m are constant, then by Lemma 5.6.1 we have that $d(x_0, x_m)$ is $\Theta(\sum_{0 \leq i < m} d(x_i, x_{i+1}))$. Thus, Lemma 5.6.1 is sufficient to show that the upper bounds derived in Chapter 3 using the triangle inequality hold to within a constant factor given only a weakly λ-approximate metric. Similarly, the upper bounds derived Chapter 4 and in this chapter using the λ-approximate triangle inequality on constant-length sequences of points also hold to within constant factors given only a weakly λ-approximate metric. (Remark: Lemma 5.6.2 is needed since our online median algorithm is used as a subroutine in the algorithms presented in Chapter 4.) However, in Lemmas 5.4.2, 5.4.10, and 5.5.1 we derive upper bounds on distances by applying the λ-approximate triangle inequality to non-constant length sequences of points that appear in the sequences σ_i associated with our online median algorithm. In these cases, the nonconstant-length sequences of points we consider have the property that they are composed of a constant number of contiguous subsequences in which distances between successive points are either geometrically increasing or geometrically decreasing. Lemma 5.6.2 below shows that the upper bounds derived using these sequences hold to within a constant factor assuming only a weakly λ-approximate metric.

Lemma 5.6.2 *Let d be a weakly λ-approximate metric, and let $x_0, x_1, \ldots, x_m$ be points such that for $1 \le i \le m$, $d(x_i, x_{i+1}) \le d(x_{i-1}, x_i)/\xi$ for a positive real $\xi > \lambda$. Then, $d(x_0, x_m) \le \frac{\lambda\xi}{\xi-\lambda} d(x_0, x_1)$.*

Proof: We first prove by induction that $d(x_0, x_m) \le \sum_{0 \le i < m} \lambda^{i+1} d(x_i, x_{i+1})$. For the base case, take $m = 1$. Then, $d(x_0, x_1) \le \lambda d(x_0, x_1)$ since $\lambda \ge 1$. For the induction step, assume that for any sequence of points y_0, ..., y_i, $1 \le i < m$, $d(y_0, y_i) \le \sum_{0 \le j < i} \lambda^{j+1} d(y_j, y_{j+1})$. Observe that

$$\begin{aligned} d(x_0, x_m) &\le \lambda \left(d(x_0, x_1) + d(x_1, x_m)\right) \\ &\le \lambda d(x_0, x_1) + \lambda \left(\sum_{1 \le i < m} \lambda^i d(x_i, x_{i+1}) \right) \\ &\le \sum_{0 \le i < m} \lambda^{i+1} d(x_i, x_{i+1}), \end{aligned}$$

where the first step follows from the weakly λ-approximate triangle inequality, and the second step follows from the induction hypothesis. Then,

$$\begin{aligned} d(x_0, x_m) &\leq \sum_{0 \leq i < m} \lambda^{i+1} d(x_i, x_{i+1}) \\ &\leq \sum_{0 \leq i < m} \frac{\lambda^{i+1}}{\xi^i} d(x_0, x_1) \\ &\leq \frac{\xi\lambda}{\xi - \lambda} d(x_0, x_1), \end{aligned}$$

where the second step follows from the assumption that $d(x_i, x_{i+1}) \leq d(x_{i-1}, x_i)/\xi$ for $0 \leq i < m$, and the third step follows from the assumption that $\xi > \lambda$. ■

As stated above, Lemma 5.6.2 is needed in addition to Lemma 5.6.1 to show that the upper bounds derived in Lemmas 5.4.2, 5.4.10, and 5.5.1 hold to within a constant factor given only a weakly λ-approximate metric. We now explain how Lemmas 5.6.1 and 5.6.2 may be used to show that the upper bound obtained in part (i) of Lemma 5.4.2 holds to within a constant factor given a weakly λ-approximate metric. Recall that in part (i) of Lemma 5.4.2, we derive an upper bound on the distance $d(y', z_i)$. For the arugment, we apply the λ-approximate triangle inequality to the sequence of points $\langle y', \ldots, y, x, \ldots, z_i \rangle$ and show that $d(y', z_i)$ is within a constant factor of the sum of the distances between successive points in this sequence. The prefix $\langle y', \ldots, y \rangle$ of this sequence appears in the sequence of balls σ_j associated with our online median algorithm. By the definition of our online median algorithm, the distances between successive points in $\langle y', \ldots, y \rangle$ decrease by a factor of β. Since β and λ are constants, and since $\beta > \lambda$, we can apply Lemma 5.6.2 with $\xi = \beta$ to conclude that $d(y', y)$ is within a constant factor of the sum of distances between successive points in $\langle y', \ldots, y \rangle$ given only a weakly λ-approximate metric. By a similar application of Lemma 5.6.2 to $d(x, z_i)$ with $\langle x, \ldots, z_i \rangle$ as the sequence of points, we can conclude that $d(x, z_i)$ is within a constant factor of the sum of distances between successive points in $\langle x, \ldots, z_i \rangle$ given only a weakly λ-approximate metric. With upper bounds on $d(y', y)$ and $d(x, z_i)$, we can then apply Lemma 5.6.1

to the constant-length sequence $\langle y', y, x, z_i \rangle$ to conclude that, given only a weakly λ-approximate metric, $d(y', z_i)$ is within a constant factor of the sum of distances between successive points in the sequence $\langle y', \ldots, y, x, \ldots, z_i \rangle$. Using Lemmas 5.6.1 and 5.6.2 in this manner, the bounds derived in part (iii) of Lemma 5.4.2 and in Lemmas 5.4.10 and 5.5.1 can also be shown to hold to within constant factors given only a weakly λ-approximate metric.

Chapter 6

Experimental Work

In Chapters 3, 4 and 5, we presented simple and fast approximation algorithms for the facility location, k-median, and online median problems. In analyzing these algorithms, we showed that each was guaranteed to produce a solution within a constant factor of optimal. In this chapter, we study the performance of Java implementations of our k-median and online median algorithms. We compare their performance to that of the widely-used k-means heuristic on a variety of synthetically generated inputs. We also apply our online median algorithm to the problem of particle recognition in electron microscopy images.

6.1 Discussion of Implementation

We implemented our uniform weights k-median algorithm and our online median algorithm in Java. For the purposes of comparison, we also implemented a fairly standard version of the k-means heuristic that works as follows. Given a set of n points, we first take the mean $\vec{m}$ of the n points. Our initial solution of k points consists of k random perturbations (in every dimension) of $\vec{m}$. We will refer to this initialization procedure as the *centroid-based initialization procedure.* We then run the k-means iterative improvement step (see Section 2.1 for further discussion) as long as the improvement in solution cost at least 1% of the current cost. (Remark: We observed in our experiments that k-means typically took under 10 iterations to satisfy this stopping condition.) Each iteration of our implementation of k-means has an asymptotic running time of $O(nkd)$, where d is the dimensionality of the points in the input. Although our algorithms are applicable in any *metric space*, all of our implementations were tested with inputs containing Euclidean points. We also implemented a data structure for the storage of sparse Euclidean points that was used in all three implementations. (We note that our k-median and online median implementation are actually implemented using an abstract point set class so that they can be used without modification with more general data structures.)

We note that our implementations were relatively concise. Our implementation of a point set data structure required 542 lines of code. Our implementation of the k-means heuristic only took 218 lines of code. In comparison, the implementation of our k-median algorithm required 726 lines of code. The implementation of our online median algorithm took 800 lines of code.

Recall that the running times derived in Chapters 4 and 5 assumed that the interpoint distances were available in $O(1)$ time. Since we apply our algorithms specifically to the case of Euclidean points, we restate the time bounds of our algorithms with respect to the Euclidean case. Our uniform-weights k-median algorithm runs in $O(nk)$ time for $k \geq \log n$ if the distances are given in the input. Since the distance between two d-dimensional Euclidean points can be computed in $O(d)$ time, the running of our k-median algorithm is then $O(nkd)$ for $k \geq \log n$. Similarly, the running time of our online median algorithm is $O(n^2d + n\ell d)$ for the case of Euclidean points.

6.2 Experiments with Synthetic Inputs

We performed two experiments with our k-means and k-median implementations. The first is a *scalability experiment* that is designed to test the perfomance of our algorithms as the number of points is increasing. The second experiment is a *dimensionality experiment*; it is designed to test the performance of our algorithms when the dimensionality of the input is increasing. For both experiments, we compare the running time and solution cost of k-means with centroid-based initialization, and k-means heuristic with our k-median algorithm as an initialization procedure. For both implementations, we measure the cost of a solution to be the sum, over all points x, of the weight of x times the distance from x to its closest point in the solution. For an alternate assessment of solution quality, we also report the average distance to the nearest cluster center. The purpose of these experiments is to assess the practicality of our k-median algorithm as an initialization procedure for k-means.

6.2.1 Methodology

For both experiments, we use randomly generated mixtures of Gaussians as the inputs. As mentioned above, we perform experiments for a variety of input parameters. Given the desired number of d-dimensional points and number of Gaussians c, we generate the inputs as follows. For each Gaussian, we randomly generate a d-dimensional point and take it as the mean of the Gaussian. Then, we generate $\lfloor \frac{n}{c} \rfloor$ unit-weight points associated with this Gaussian as follows. Given the desired radius r of the Gaussian, we draw d random numbers independently from a Gaussian distribution with mean 0 and standard deviation r and add the resulting d-dimensional vector to the mean. Repeating this procedure $\lfloor \frac{n}{c} \rfloor$ times yields a d-dimensional Gaussian with $\lfloor \frac{n}{c} \rfloor$ points (i.e., mixing weight roughly $\frac{1}{c}$). To ensure that there are exactly n points in the input, we generate $n - (n-1) \lfloor \frac{n}{c} \rfloor$ points for

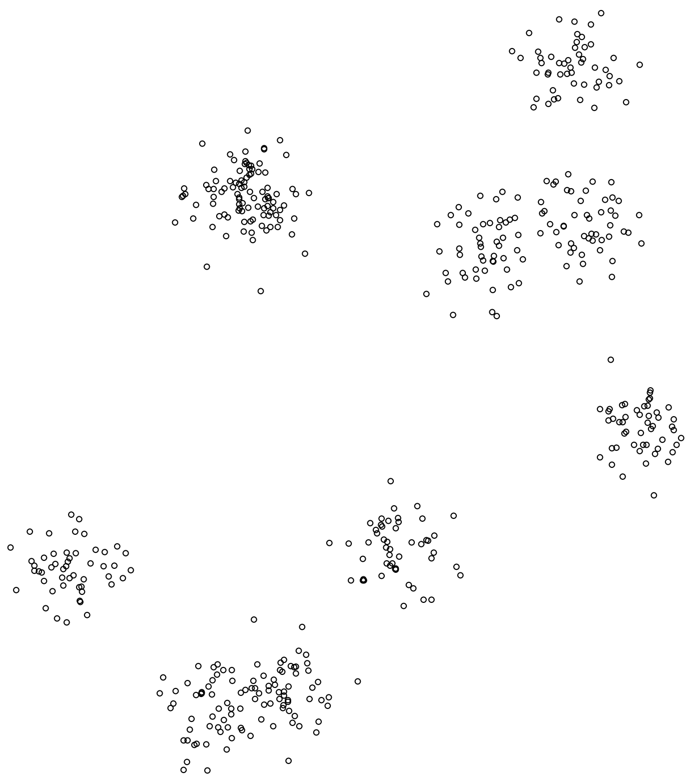

Figure 6.1: A mixture of 10 2-dimensional Gaussians with $n = 500$.

the last Gaussian. We make use of a single value of r for all of our experiments. For an example of our synthetically generated inputs, we refer the reader to Figure 6.1. For our experiments, we set the α and β parameters for our k-median algorithm to 1 and .5, respectively. We ran our experiments on an Intel Pentium 4 workstation with a clock speed of 996 megaHertz running Debian Linux 2.4.10 with 2 gigabytes of main memory. For all of our experiments, we used Java 1.3.1 compiler with the `-O` switch and the Java 1.3.1 interpreter with the `-hotspot` optimization enabled.

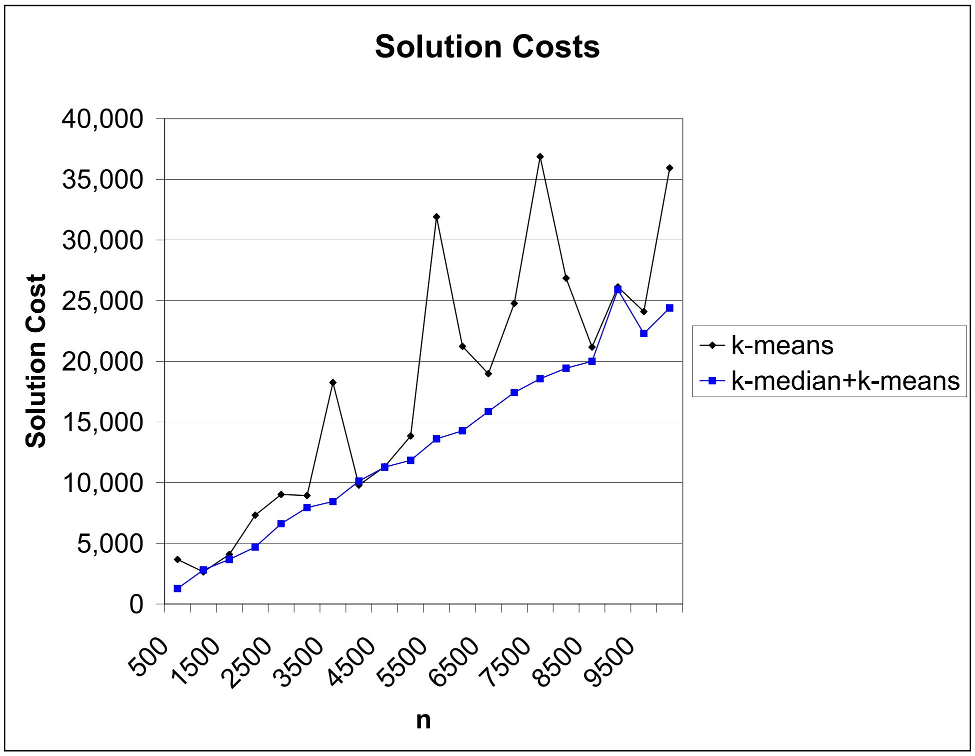

Figure 6.2: Solution costs for our scalability experiment.

The minimum and maximum heap sizes for the interpreter were both set to 600 megabytes.

6.2.2 Scalability Experiment

For our scalability experiment, we ran our implementations with test inputs with $c = 10$, $d = 2$, and n ranging from 100 to 10,000 in increments of 500. For k-means and our k-median implementations, we set the parameter k equal to c. The running times of our implementations is given in Figure 6.3, the solution costs are reported in Figure 6.2, and the average distances to the nearest cluster center are given in Figure 6.4. The results of this experiment show that the approximation guarantees of our k-median algorithm are noticeable for the synthetic inputs we consider. When

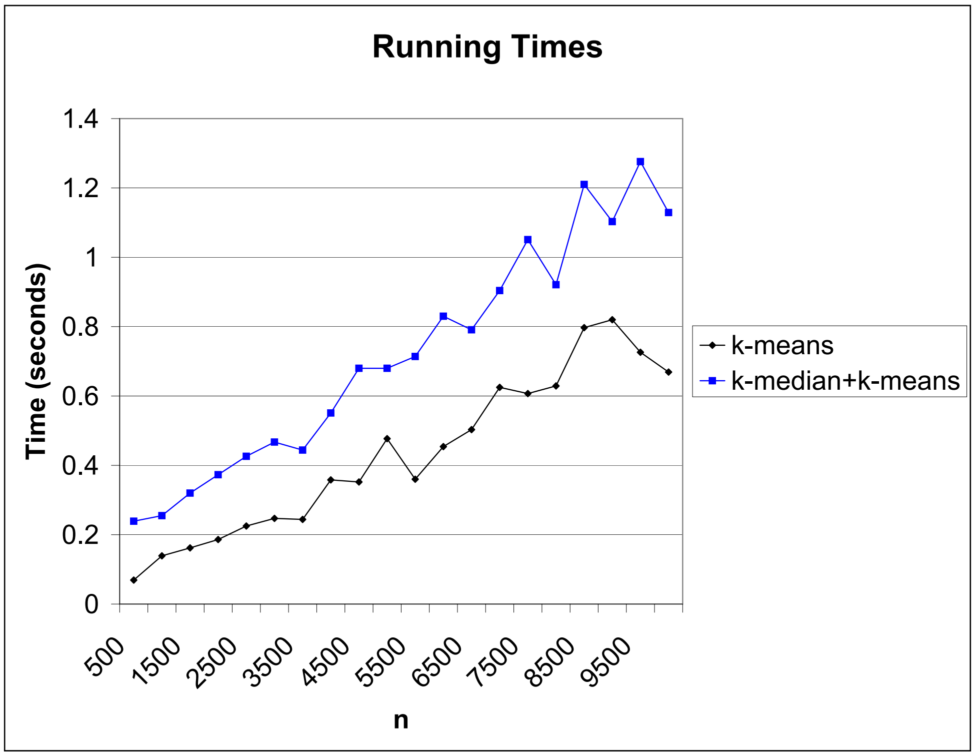

Figure 6.3: Running times for our scalability experiment.

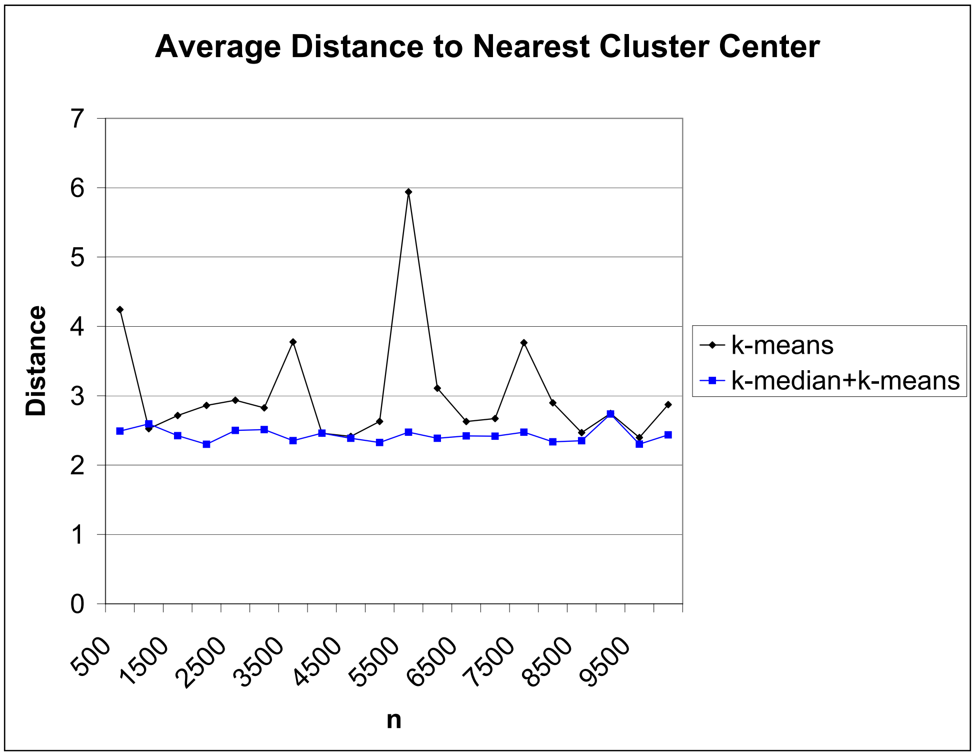

Figure 6.4: Average distance to the nearest cluster center for our scalability experiment.

our k-median algorithm is used as an initialization procedure, the resulting solution costs were, on average, roughly 40% lower than those produced by the centroid-based k-means implementation. Similarly, the average distance to the nearest cluster center is roughly 25% lower when our k-median algorithm is used as an initialization procedure. In contrast, the running time of k-means with the centroid-based initialization procedure was about 40% faster than with our k-median algorithm as the initialization procedure. We note however that our approach is qualitatively still quite fast. When our k-median algorithm is used as the initialization procedure, the slowest running time was just 1.13 seconds for $n = 10,000$.

6.2.3 Dimensionality Experiment

For our dimensionality experiments, we considered inputs with $c = 10$, $n = 500$, and d ranging from 100 to 1000 in increments of 100. As in the scalability experiment, we set the parameter k equal to c. The running times of our implementations are given in Figure 6.6, and the associated solution costs are given in Figure 6.5 The results of our dimensionality experiment suggest that our using our k-median implementation as an initialization procedure is still beneficial. As in our scalability experiment, we noticed that when our k-median algorithm is used as an initialization procedure, the resulting solution costs are much lower. In this experiment, using our k-median algorithm to initialize k-means yielded solution costs that were, on average, 175% lower than when the centroid-based initialization procedure was used to initialize k-means; the average distance to the nearest cluster center was about 100% lower. As in the scalability experiment, our k-median algorithm yielded slower running times; k-means with centroid-based initialization was on average only 20% faster than k-means with our k-median algorithm as an initialization procedure. Given the results of our scalability experiment, the results of our dimensionality experiment seem to indicate that the approximation guarantees of our k-median algorithm are

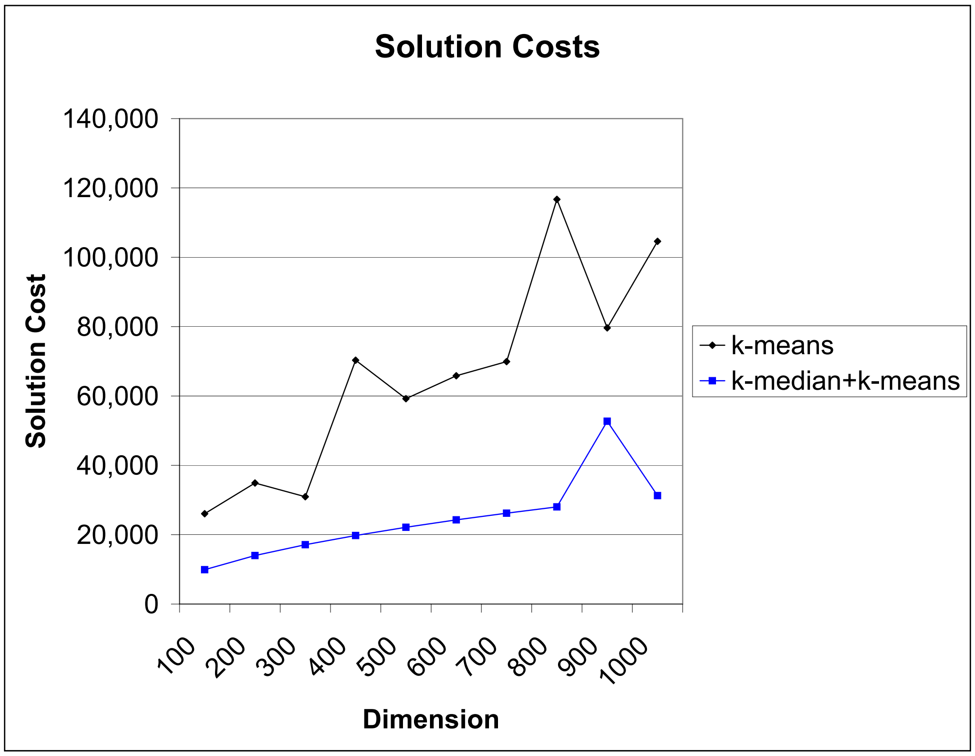

Figure 6.5: Solution costs for our dimensionality experiment.

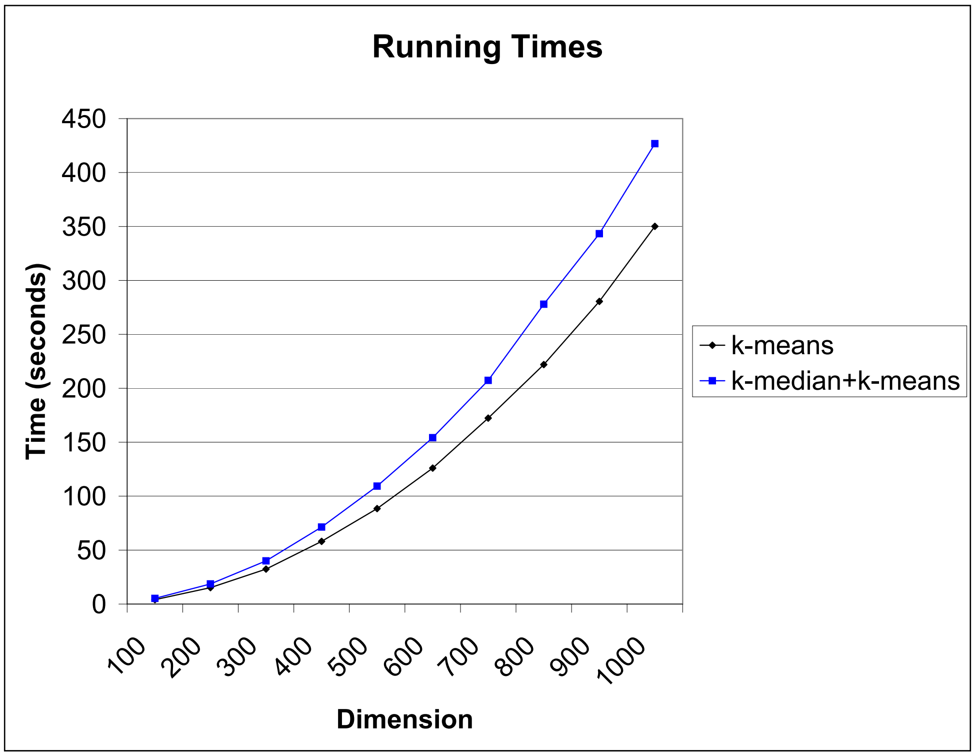

Figure 6.6: Running times for our dimensionality experiment.

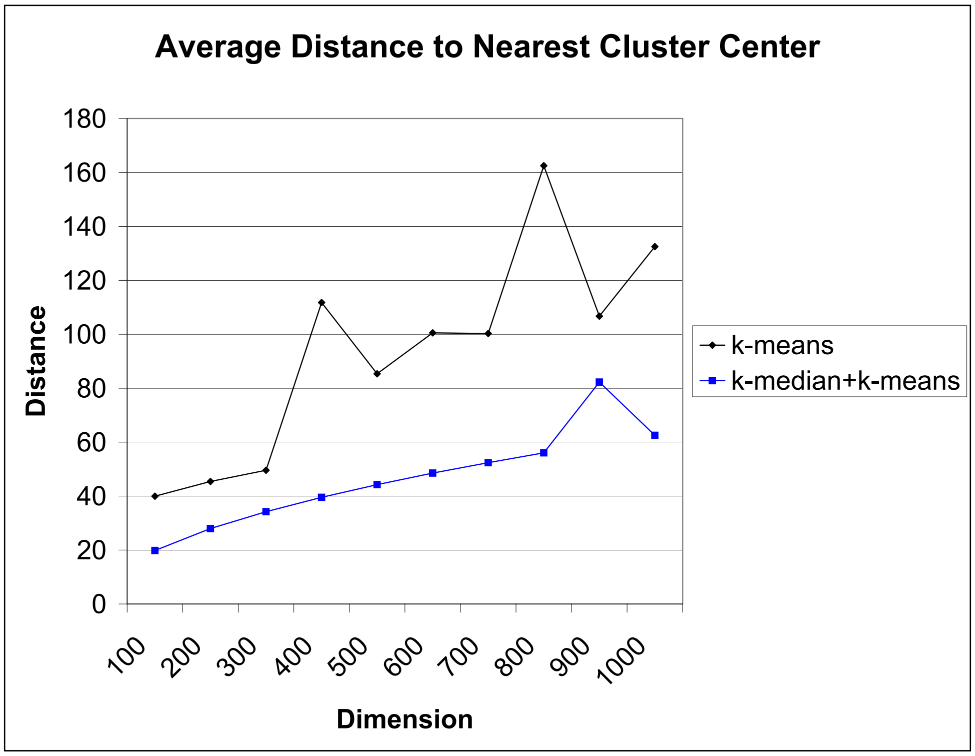

Figure 6.7: Average distance to the nearest cluster center for our dimensionality experiment.

even more useful as the dimensionality of the input is increasing.

6.2.4 Discussion

In both of the experiments above, using our k-median implementation as an initialization procedure for k-means reduced the solution cost significantly without a prohibitive time penalty. Thus, it appears that the approximation guarantees of our k-median algorithm translate to low solution costs in practice, especially for high-dimensional inputs. However, these reduced solution costs are accompanied by increased running times for the k-means heuristic with a k-median initialization procedure. Although our k-median algorithm has the same asymptotic running time (for a wide range of value of k) as just one iteration of k-means, this gap is likely due to the fact that the centroid-based initialization procedure has a running time of just $O(n + dk)$.

It would be interesting to see if the overall gap in running times can at least be made smaller with improvements to the implementation. For example, there is still much room for optimization of the α and β parameters of our k-median algorithm. We note that we only lowered the α parameter to a reasonable value with a few experiments. With more empirical data, it may be possible to find values of α and β that yield lower running times without a significant increase in solution cost. A more sophisticated optimization would be to make both the α and β parameters adapt to the number of remaining points in each iteration of the successive sampling algorithm. Additionally, our k-median algorithm uses much more memory than the simple centroid-based initialization procedure. Thus, with an optimized implementation of our k-median algorithm in another language, for example, C++, that has significantly less memory overhead, it may be possible to obtain more competitive running times.

6.3 Clustering in Electron Microscopy

In this section, we explore the applicability of our online median algorithm to the problem of *particle selection* in electron microscopy. Given a two-dimensional electron microscopy image, the problem of particle selection asks us to identify the location of the particles present in the source of the image. The main difficulty in attacking this problem is the characteristically low signal-to-noise ratio of the images obtained from electron microscopy. However, accurate algorithms for this problem have significant practical application, since, for example, algorithms for particle selection in two dimensions are crucial to determining the three-dimensional reconstruction of electron microscopy images [48]. This problem was first studied by [45] and has subsequently been studied by many researchers(see, e.g., [12, 17, 29, 37, 41, 44], and [11] for further references). Typical approaches to this problem make use of heuristics, although there are also approaches that make use of edge-detection algorithms [17].

For the particle selection problem, although the approximate size of the particles is known, it is common that the number of particles in the given input image is unknown. Given the low signal-to-noise ratio of the images, it may be desirable to view the image at various levels of granularity to determine the number and location of the particles. As mentioned in Chapter 5, our online median algorithm could be of practical use for such a problem. To study the performance of our online median algorithm for the purpose of particle selection, we applied our implementation to several sample images obtained from electron micrographs. These images were obtained from the experiments of Yu and Bajaj [48].

6.3.1 Results

We ran our online median implementation on four test images, each of which was simply a set of weighted two-dimensional points. We set the α parameter of our online median algorithm to 3.732. This value was chosen since it is approximately

$2+\sqrt{3}$, the value of α that minimizes the competitive constant of our online median algorithm (see Section 5.1 for further discussion). For each image, we computed the the twenty cluster centers of an online median ordering for a set of points associated with each image. We examined our results to see if the cluster centers obtained corresponded to possible particles in the source image and guessed the number of particles in the image by inspection. We note that this procedure could be partially automated by using existing heuristics to estimate the number of clusters (see e.g. [10] for pointers to the literature). Figures 6.8, 6.10, 6.12, and 6.14 show the four electron microscopy images that we worked with. For each image, we obtained a set of weighted two-dimensional points from the work of [48] which we used as an input to our online median algorithm. These inputs were obtained by thresholding the pixels in the source image; this thresholding is based on the minimum and maximum intensities that occur in the image (see [48] for further discussion). For reference, we will label the images as Image 0, 1, 2, and 3 and the inputs associated with Image 0, 1, 2, and 3 as Input 0, 1, 2, and 3, respectively. Input 0 contains 376 points, input 1 contains 828 points, and inputs 2 and 3 each contain 822 points. Figures 6.9, 6.11, 6.13, and 6.15 show these four inputs. In each figure, points are represented by circles whose area is proportional to the point weight.

Figure 6.16 shows the manner in which our online median implementation successively adds cluster centers for Input 0. Each cluster center is indicated by a box whose lower left corner is the location of the chosen cluster center. The number inside the box indicates when a given cluster center was chosen. For example, the lower left corner of the box labelled 0 shows the location of the first cluster center chosen by our online median implementation. Our results are below in Figures 6.17, 6.19, 6.21, and 6.23. For each image, we chose the number of cluster centers interactively. The cluster centers are numbered in the order in which they were computed by our online median implementation. For comparison, the results of Yu and Bajaj [48] are

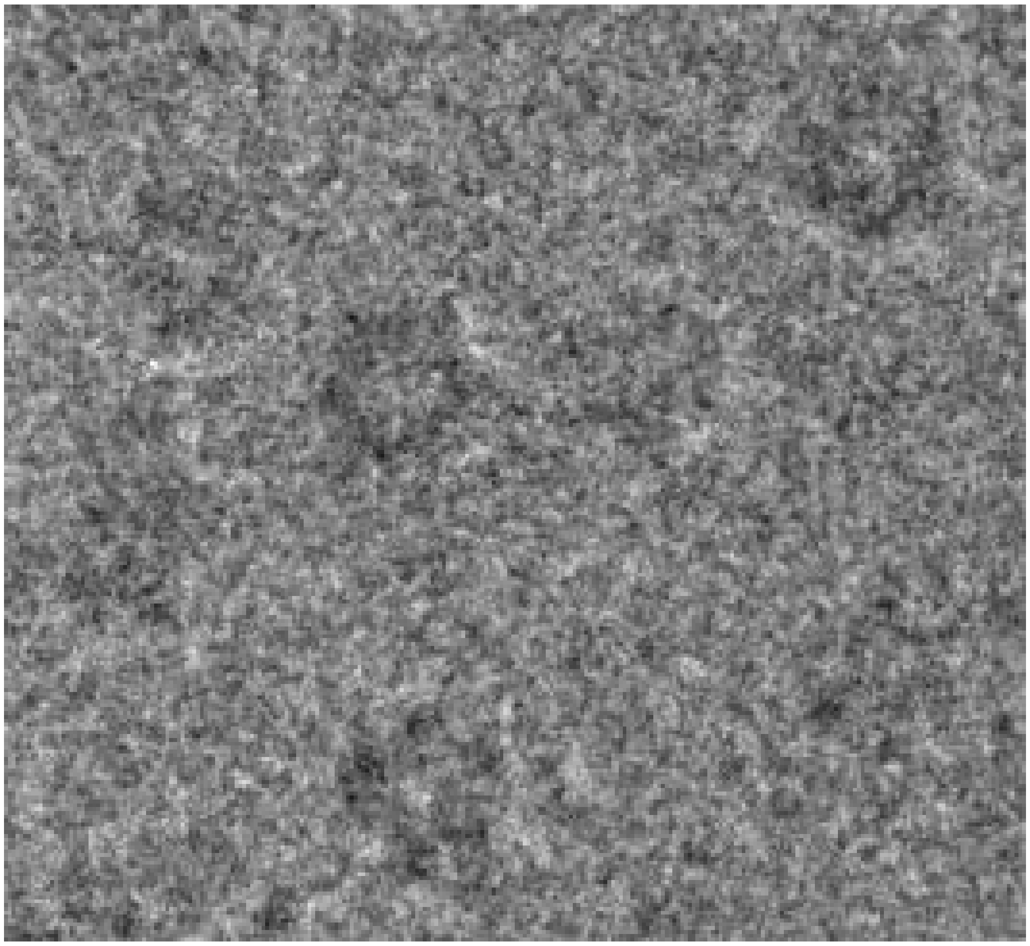

Figure 6.8: Image 0

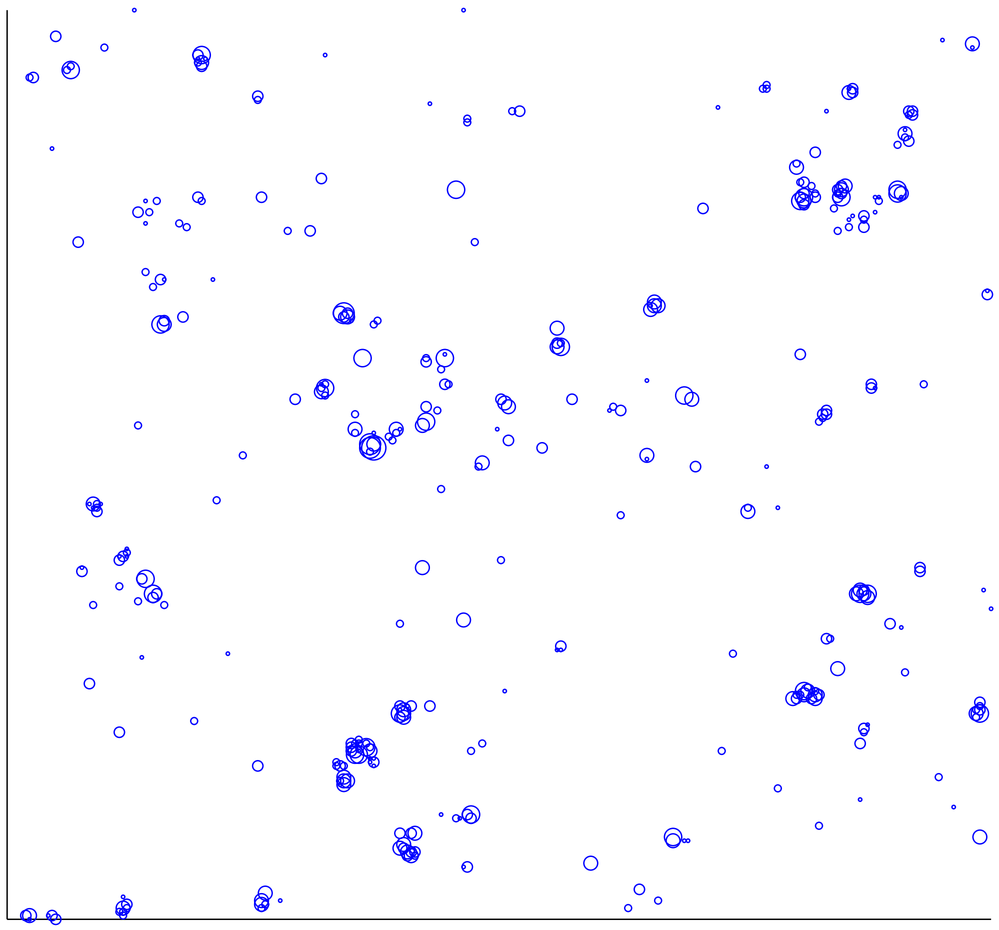

Figure 6.9: Input 0

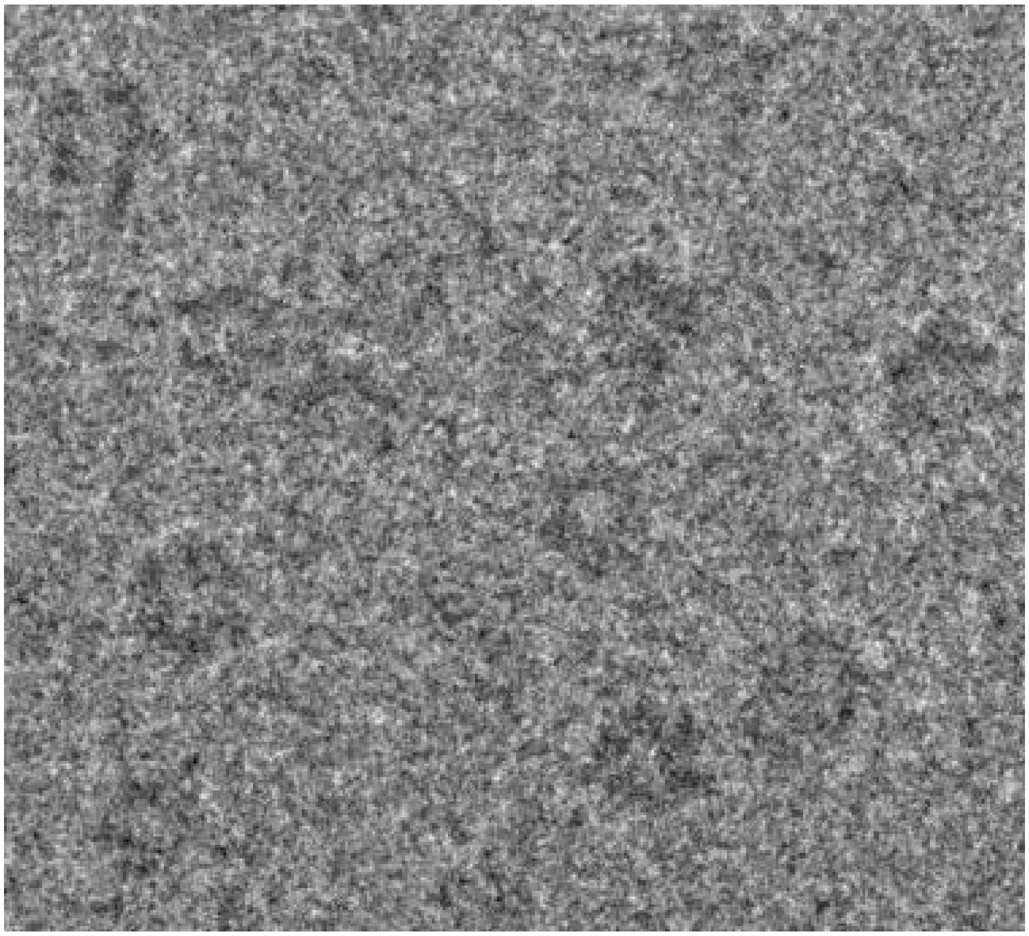

Figure 6.10: Image 1

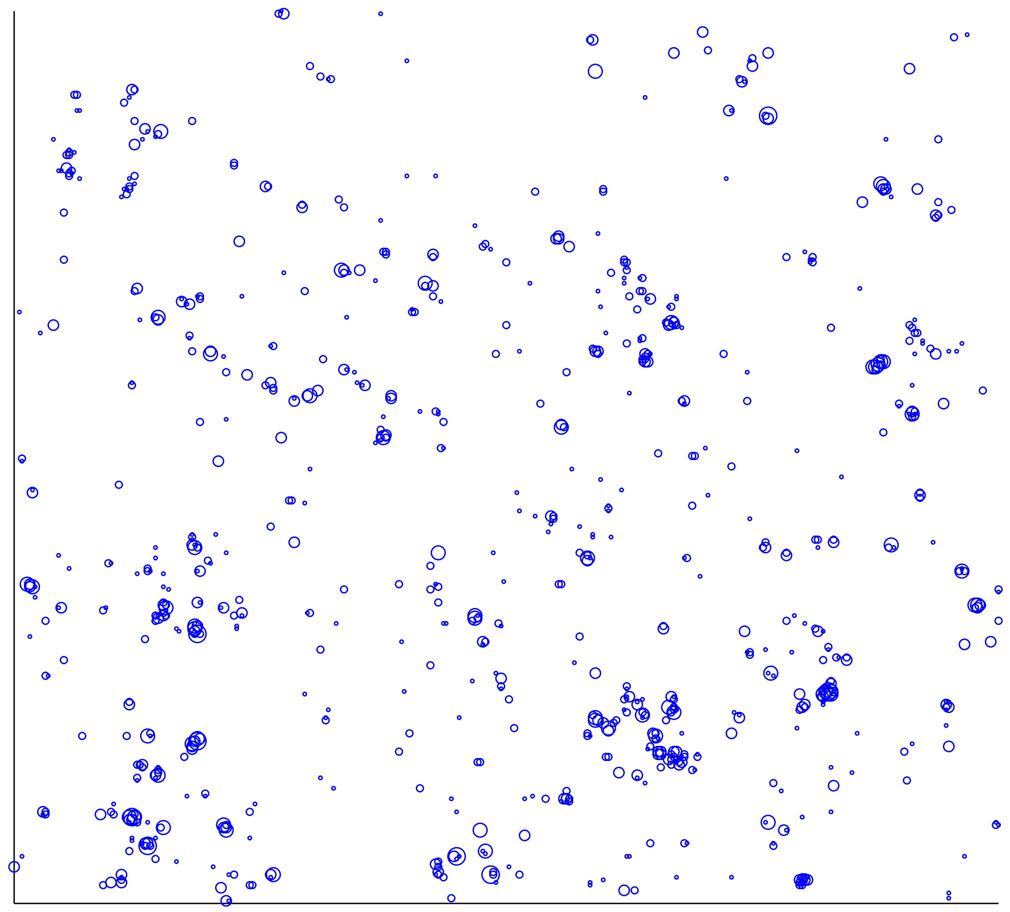

Figure 6.11: Input 1

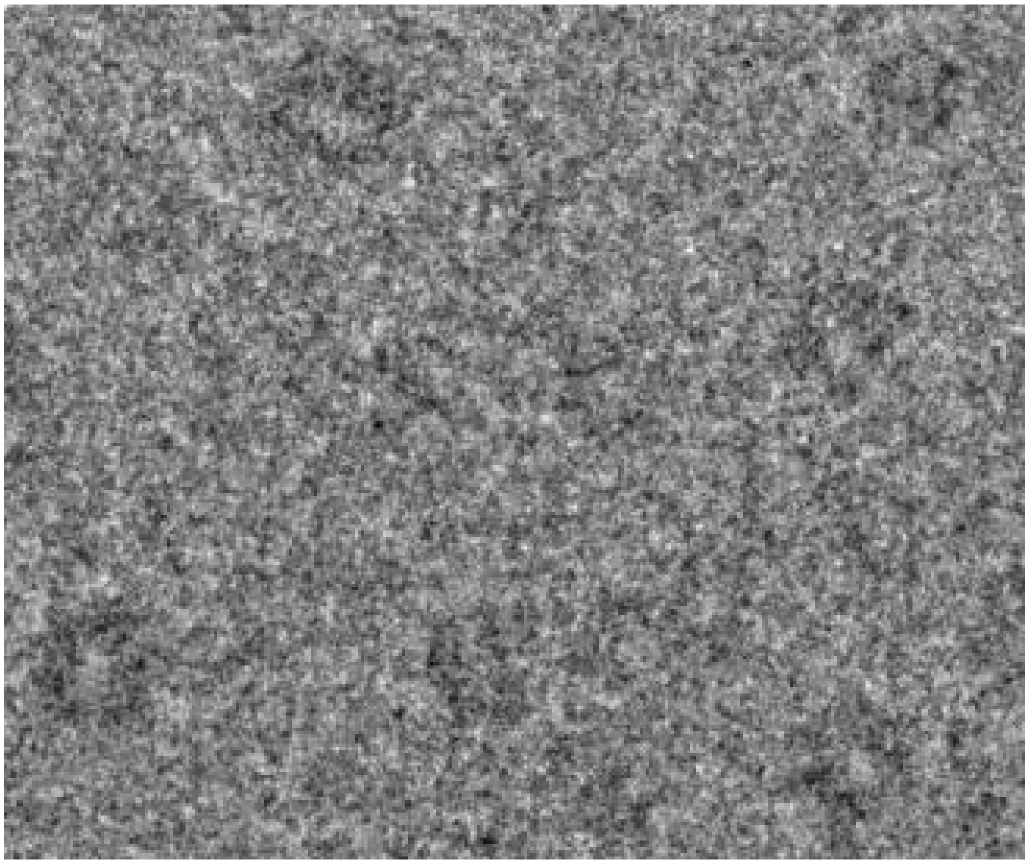

Figure 6.12: Image 2

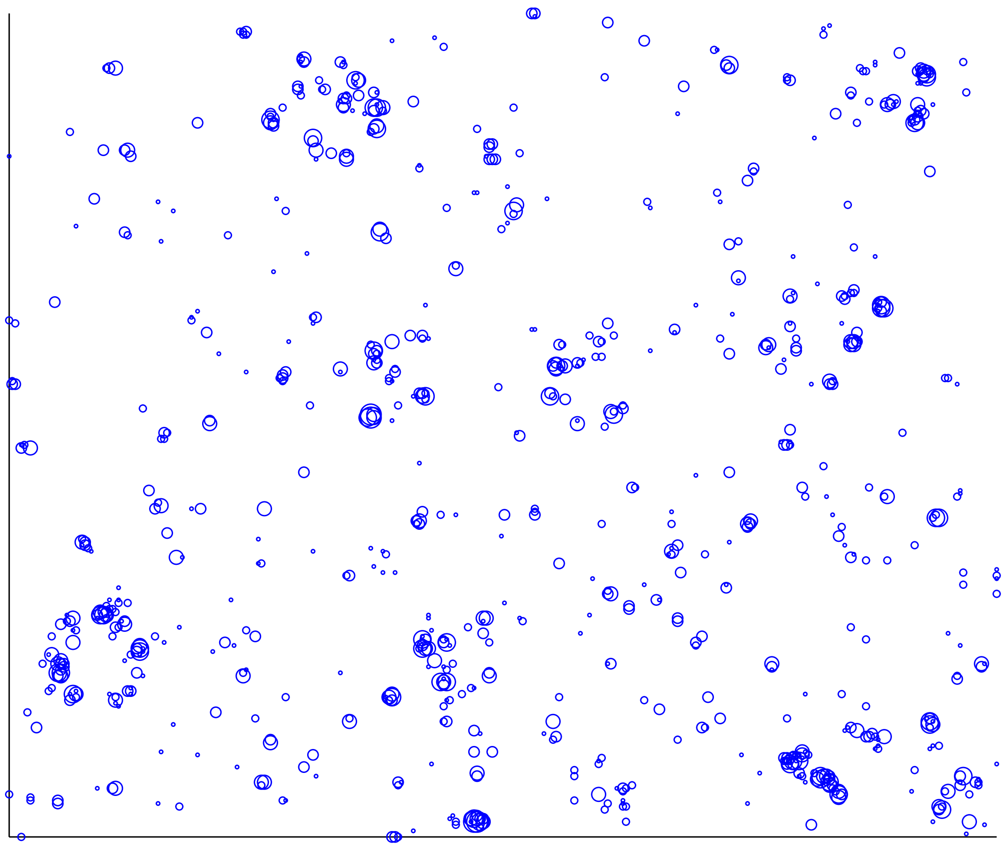

Figure 6.13: Input 2

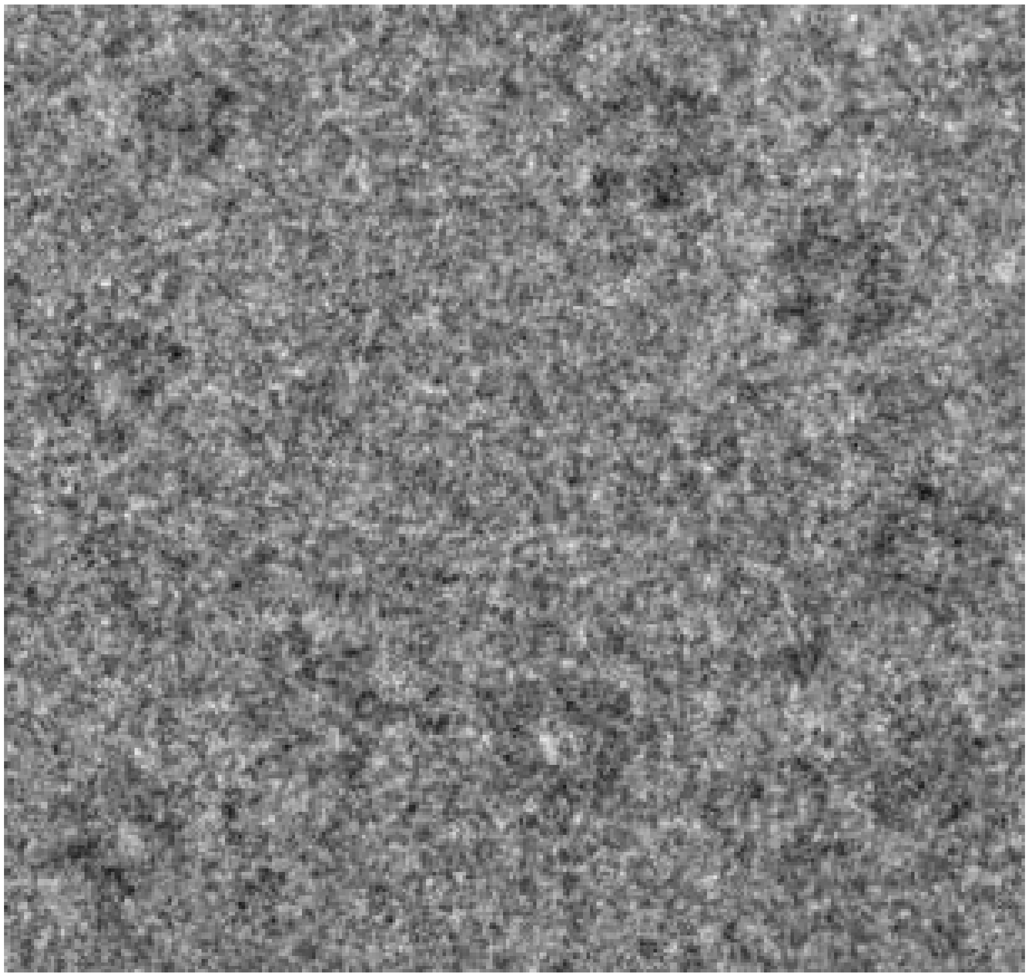

Figure 6.14: Image 3

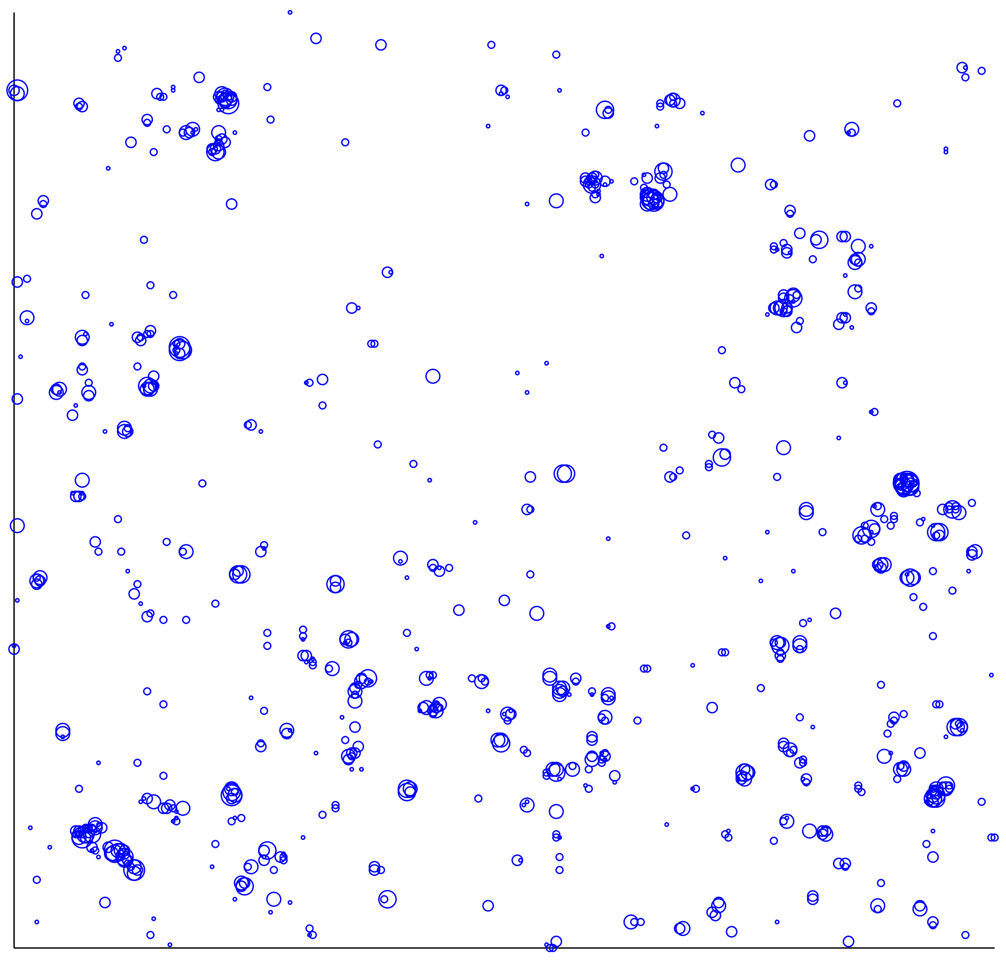
Figure 6.15: Input 3

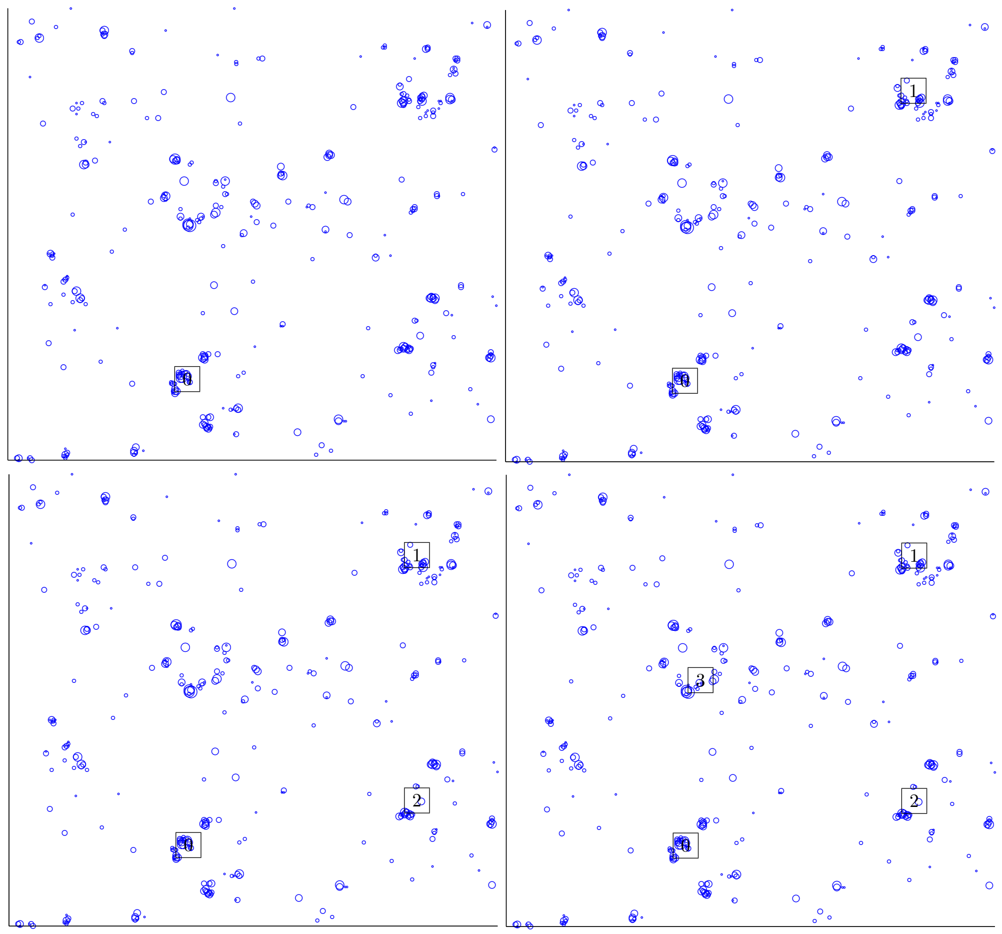

Figure 6.16: A sample execution of our online median implementation for Image 0.

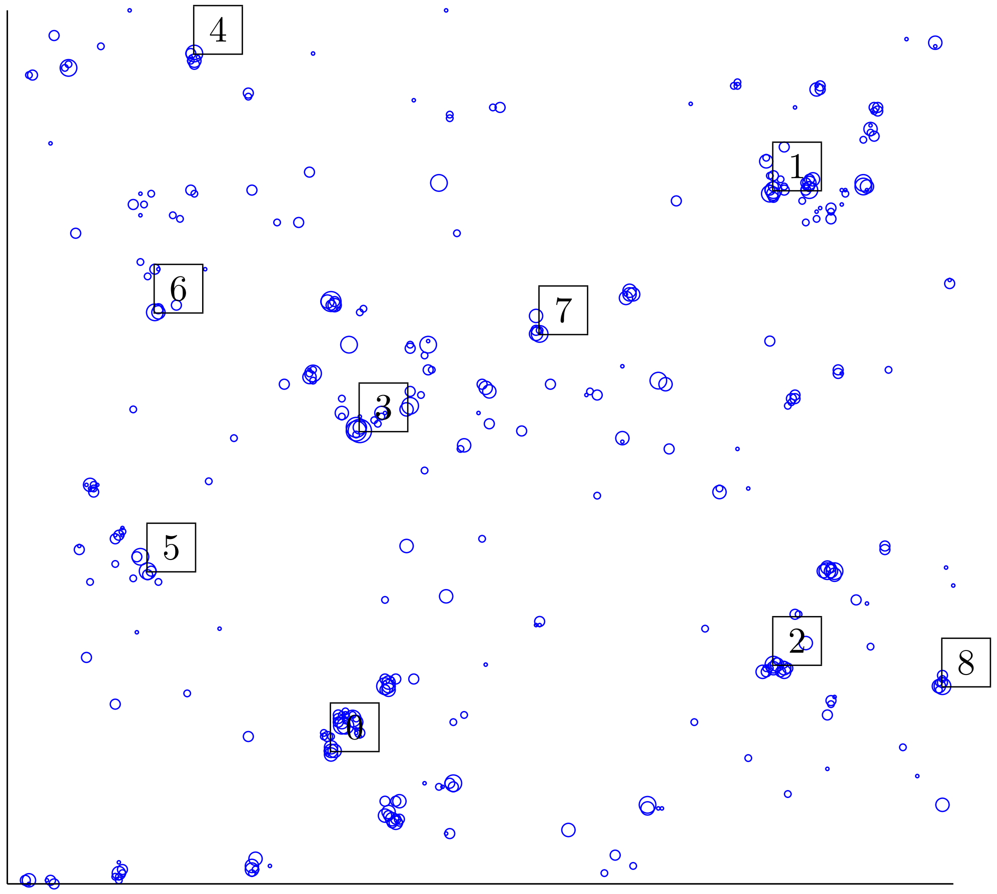

Figure 6.17: Results for Image 0.

in Figures 6.18, 6.20, 6.22, and 6.24 for Images 0 through 3. In these figures, the selected particles are indicated by white boxes. We note that Bajaj and Yu use the same point sets as input to their particle selection technique.

6.3.2 Discussion

The results in Figures 6.17 through 6.23 seem to indicate that our online median algorithm could have practical use for the problem of particle selection. For each input, our online median algorithm identified what seemed to be particles in the

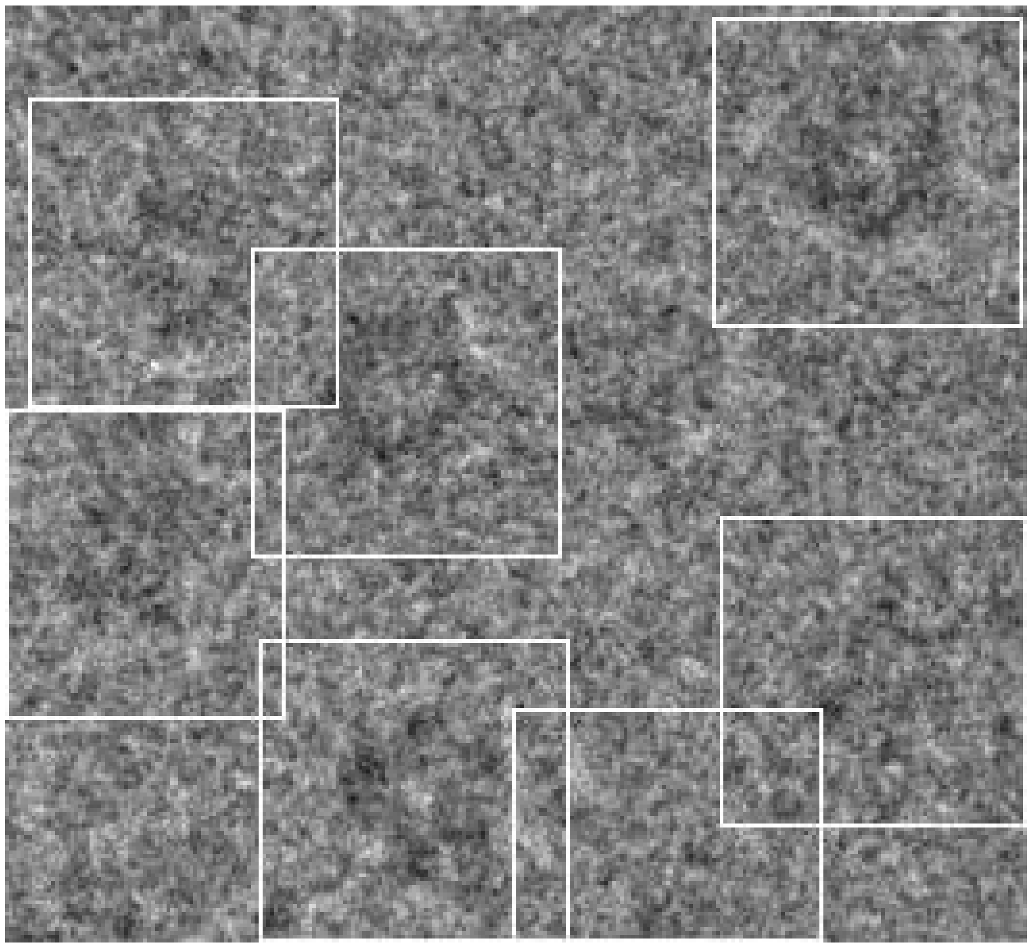

Figure 6.18: Results of Yu and Bajaj [48] for Image 0

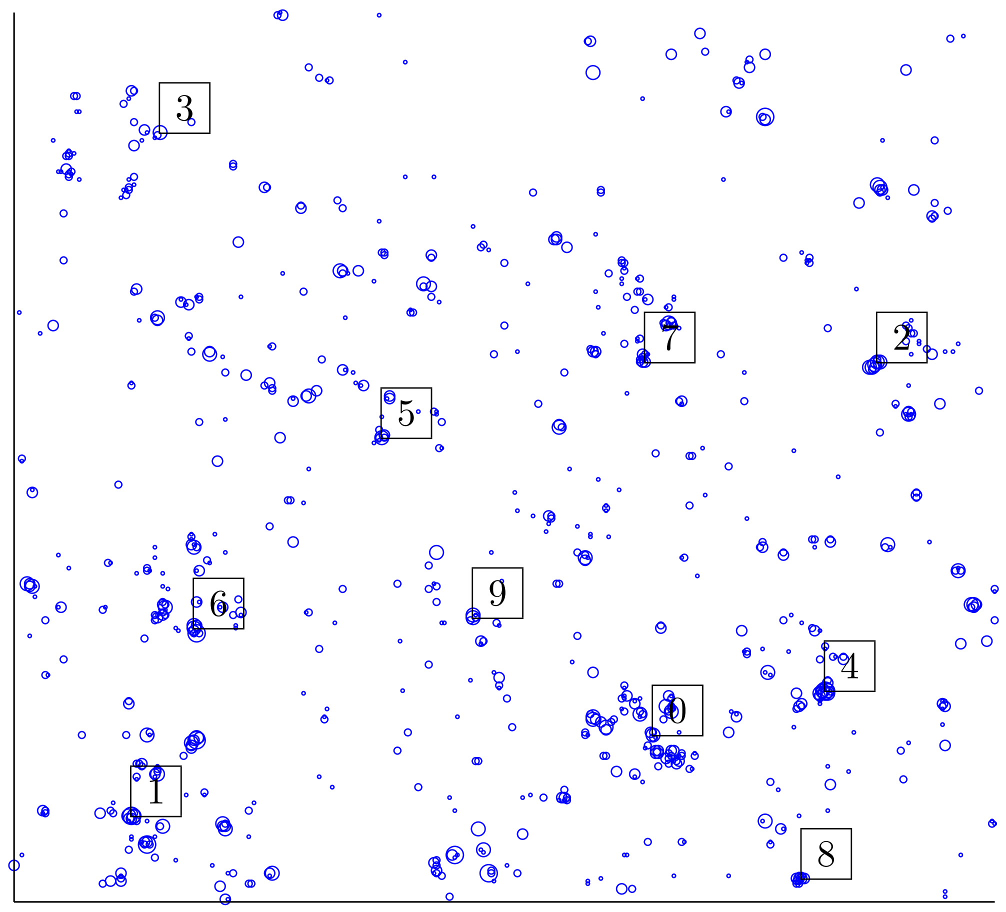

Figure 6.19: Results for Image 1

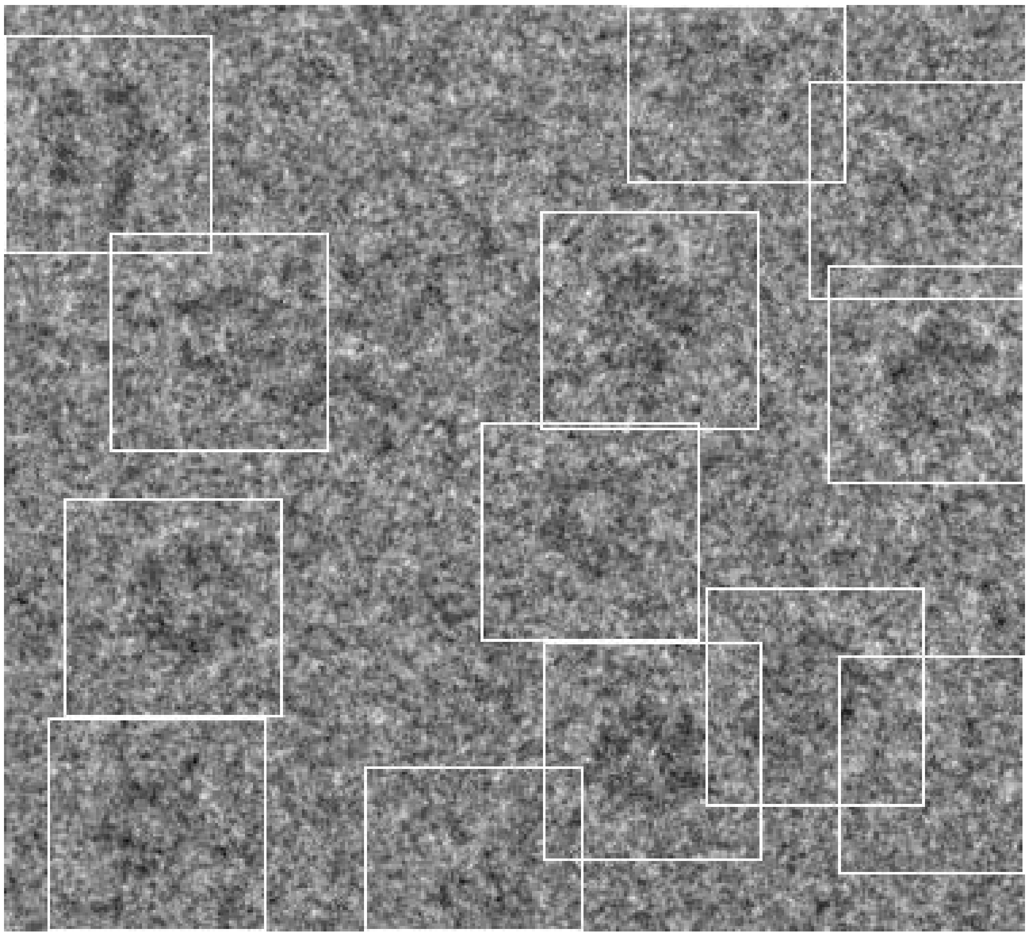

Figure 6.20: Results of Yu and Bajaj [48] for Image 1

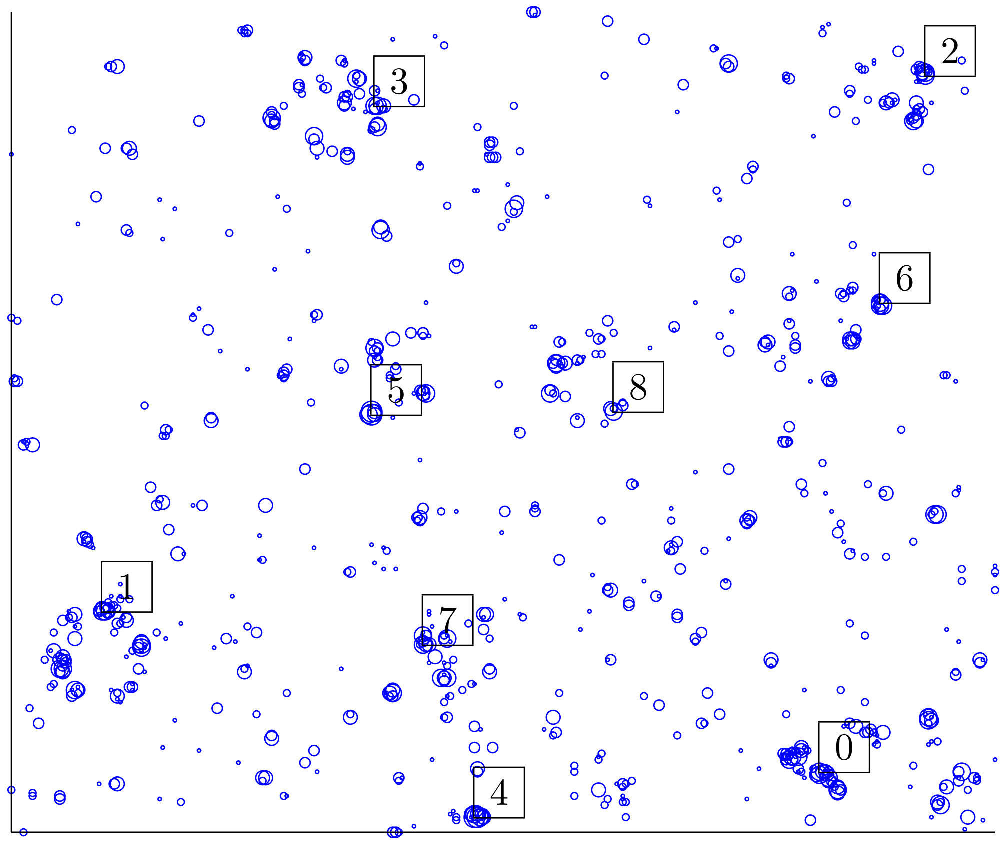

Figure 6.21: Results for Image 2

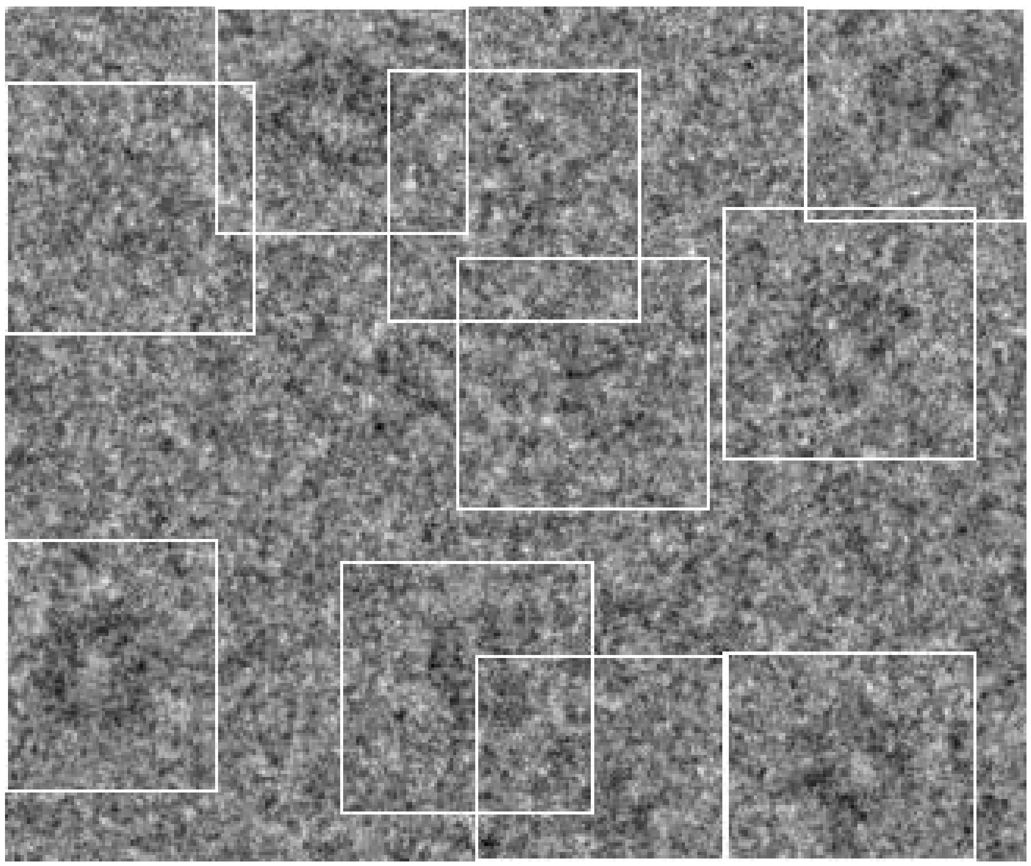

Figure 6.22: Results of Yu and Bajaj [48] for Image 2

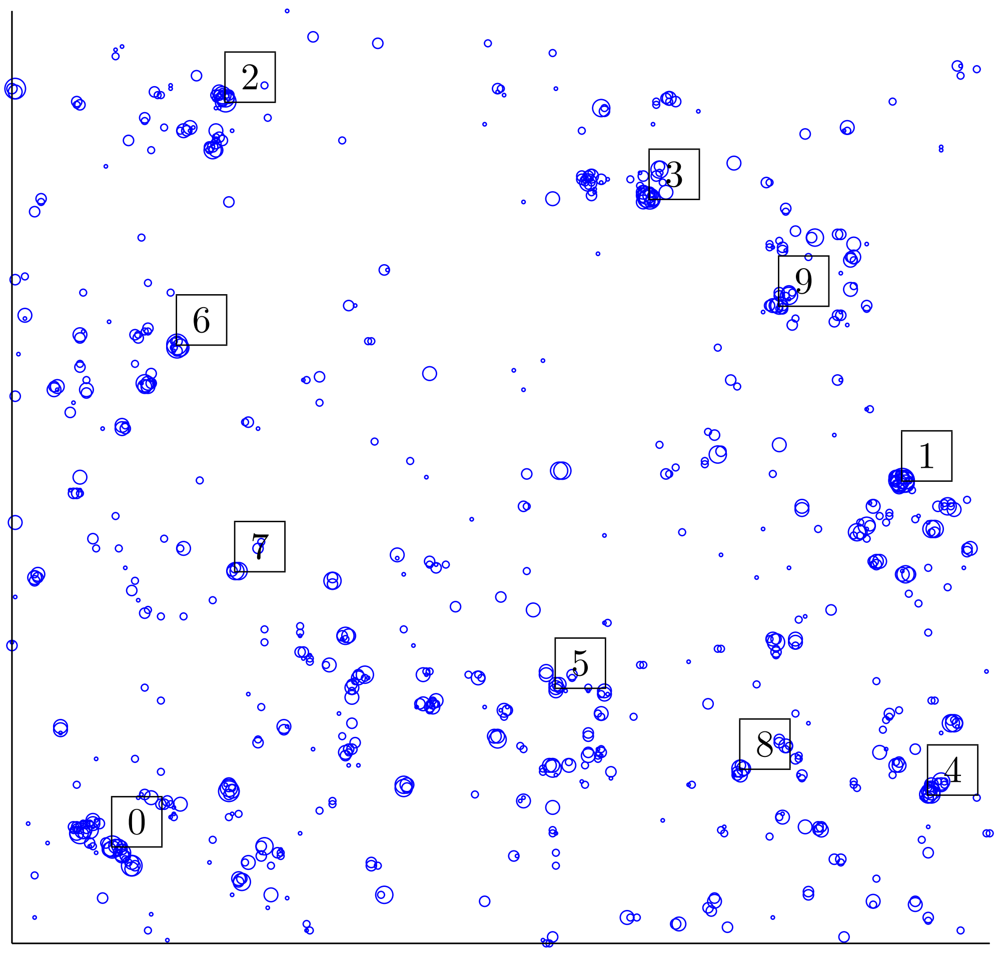

Figure 6.23: Results for Image 3

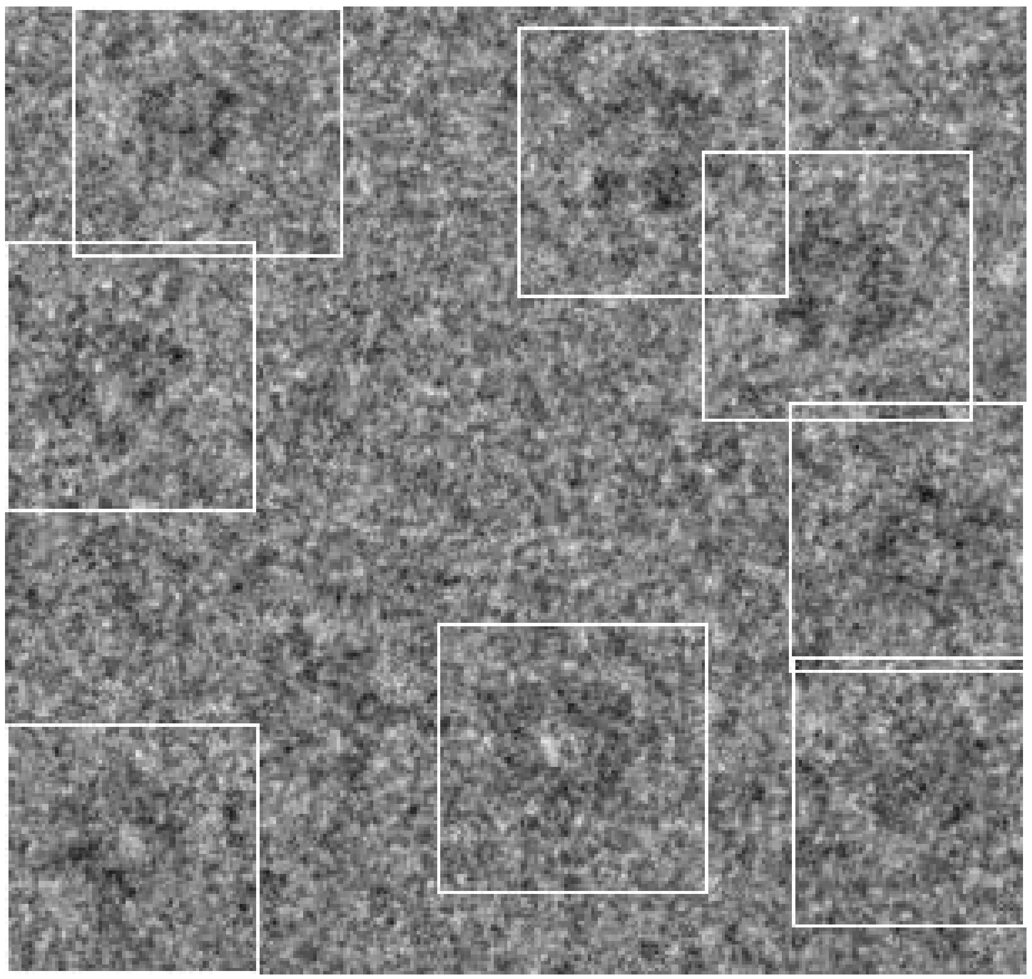

Figure 6.24: Results of Yu and Bajaj [48] for Image 3

source image. When compared to the results of Yu and Bajaj, our implementation identifies nearly the same set of particles. Qualitatively, our approach seems to identify interesting regions in all four of the inputs. Overall, our online median algorithm seemed to cope well with the low signal-to-noise ratio in the inputs.

Although we have obtained qualitatively good results, we note that our images are quite small. Our largest image was 377 pixels by 342 pixels, but in practice images could be much larger. A drawback of our algorithm is that the preprocessing time of our algorithm is quadratic in the number of input points. We note that this property of our algorithm makes it very sensitive to the thresholding procedure used to translate the source image to a point set. One simple way to cope with this limitation would be to simply split up a large source image and treat it as a number of smaller images. After the initialization procedure, however, our algorithm requires just $O(\ell)$ amortized time to identify the next cluster center. (Recall that the parameter ℓ was roughly the number of bits needed to represent distances in the input.) Thus, our algorithm would be well-suited for use in an interactive environment in which a large number of relatively small images must be processed quickly.

We note that the online property of our online median algorithm seemed to bias the choice of cluster centers (see, e.g., Figure 6.16). In our results, the cluster centers that were computed by our online median implementation did not always correspond to the center of a region that could be a particle. It would be interesting to see if we could use a postprocessing step to refine the cluster centers computed by our online median algorithm. We saw in Section 6.2 that the k-means heuristic runs very quickly even for large values of n. It would be interesting to see, for example, if the k-means heuristic could be combined with our online median algorithm to iteratively refine the cluster centers each time a new cluster center is computed.

Chapter 7

Concluding Remarks

In this dissertation, we have presented fast constant-factor approximation algorithms for the facility location, k-median and online median problems. We show that our facility location and online median algorithms are time-optimal and that our k-median algorithm is time-optimal for a wide range of values of k. Our k-median and online median algorithms are based on two algorithmic techniques, successive sampling, and the hierarchically greedy strategy, that could be of independent interest. Additionally, we show that all of our approximation results hold for objective functions other than the k-median objective function. Specifically, we show that all of our approximation results hold, to within constant factors, for the k-means objective function. We also show that the approximation guarantees of our algorithms could have significant practical benefit as well. When used as an initialization procedure for k-means, our k-median algorithm seems to achieve a very reasonable tradeoff between running time and solution quality. In addition, our online median algorithm seems to have practical benefit for the problem of particle selection in electron microscopy. Our preliminary results indicate that it may be useful in an interactive setting.

In Chapter 2 we mentioned that the facility location and k-median problem were formulated, and are still being studied, in the context of resource placement. It would be interesting to see if the hierarchically greedy strategy employed in our online median algorithm is useful for problems in resource placement on a network. For example, Korupolu *et al.* give a simple constant-factor approximation algorithm for the problem of hierarchical cooperative caching on a network. For the version of the problem they study, it is assumed that the given distance function is a *tree metric*. That is, distances between nodes are defined by a tree underlying the network. (Remark: This type of distance function is also referred to as an *ultrametric*.) The algorithm of Korupolu *et al.* is based on a variation of the naive greedy strategy; it would be interesting to see if our generalization of the naive greedy strategy could be used to obtain a constant-factor approximation algorithm for the problem of cooperative caching in an arbitrary metric space.

We showed in Chapter 4 that our successive sampling algorithm captures the essence of the input for the purpose of clustering with a very small set of sampled points (i.e., a subset of size roughly $O(k \log n/k)$). Our approach to the k-median problem was to first use our successive sampling procedure to distill the input points and then to solve a small problem instance constructed from the sampled points using existing techniques. It would be interesting to see if we could apply this two-step approach to obtain fast approximation algorithm for other $\mathcal{NP}$-hard optimization problems.

Appendix A

Algorithm Modified-Small-Space

The main goal of this section is to establish that a modified version of algorithm Small-Space of Guha *et al.* [16] is $O(1)$-approximate. Our version of algorithm Small-Space, which we refer to as Modified-Small-Space, and its analysis are used to establish the results in Sections 4.4 and 4.5. We note that the changes to the algorithm of Guha *et al.* are trivial; the discussion in this section is included for completeness only.

We now discuss the modification to algorithm Small-Space of Guha *et al.* [16] and the changes required in the analysis. In Step 2 of algorithm Small-Space of Guha *et al.* [16], ℓ $O(k)$-configurations are computed. Then, in Step 3, a weight function is constructed based on these configurations. In algorithm Modified-Small-Space, we instead compute ℓ assignments in Step 2 and use them in Step 3 to construct a weight function. Theorem 2.4 of Guha *et al.* [16] proves the approximation bound for algorithm Small-Space. In order to prove the same approximation bound for algorithm Modified-Small-Space, a slight generalization of [16, Theorem 2.3] (which is used in the proof of [16, Theorem 2.4]) is needed. The rest of their analysis, including the proof of Theorem 2.4, remains unchanged.

This section is organized as follows. We first present algorithm Modified-Small-Space. We then restate Theorem 2.4 of Guha *et al.* [16] as Theorem 6 below. We then give the required generalization of Theorem 2.3 of Guha *et al.* [16] with Lemma A.0.1 below.

We also make use of some additional definitions in this section. For any assignment τ, we define w_τ as follows: For a point x in $\tau(U)$, $w_\tau(x) = \sum_{y \in \tau^{-1}(x)} w(y)$. For any assignment τ and set of points X, we let $c_\tau(X) = \sum_{x \in \tau(U)} d(x, X) \cdot w_\tau(x)$.

Algorithm Modified-Small-Space(U)

1. Divide U into ℓ disjoint pieces, $U_0, \ldots, U_{\ell-1}$.
2. For each i, $0 \le i < \ell$, compute an assignment $\tau_i : U_i \to U_i$. Let τ be an assignment that is defined as follows: If x is in U_i, then $\tau(x) = \tau_i(x)$.
3. Let U' denote $\tau(U)$ and let w_τ be the weight function on U'.
4. Compute a k-configuration using U' as the set of points, w_τ as the weight function, and d as the distance function.

Theorem 6 (Guha *et al.* [16]) *If an (a, b)-approximate k-median algorithm is used in Step 2 of algorithm Modified-Small-Space, and a c-approximate k-median algorithm is used in Step 4 of algorithm Modified-Small-Space, then algorithm Modified-Small-Space is* $(2c(1+2b)+2b)$*-approximate.*

Lemma A.0.1 *Let the sets* U_i*,* $0 \le i < \ell$*, be a partition of* U*. Let* τ_i*,* $0 \le i < \ell$*, be assignments such that* $\tau_i(U) \subseteq U_i$ *and* $\tau_i^{-1}(U) = U_i$*. Let* τ *be an assignment that is defined as follows: for* x *in* U_i*, then* $\tau(x) = \tau_i(x)$*. Let* X *be a configuration such that* $X \subseteq \tau(U)$*. Then,*

$$c_\tau(X) \quad \le \quad \mathit{cost}(X) + \sum_{0 \le i < \ell} c(\tau_i).$$

Proof: Observe that

$$\begin{aligned}
c_\tau(X) &= \sum_{x\in\tau(U)} d(x,X)\cdot w_\tau(x)\\
&= \sum_{x\in\tau(U)} d(x,X)\left(\sum_{y\in\tau^{-1}(x)} w(y)\right)\\
&\le \sum_{x\in\tau(U)}\sum_{y\in\tau^{-1}(x)} (d(y,\tau(y))+d(y,X))\cdot w(y)\\
&= \sum_{y\in U}(d(y,\tau(y))+d(y,X))\cdot w(y)\\
&= c\,(\tau)+\mathit{cost}\,(X)\\
&= \mathit{cost}\,(X)+\sum_{0\le i<\ell} c\,(\tau_i)\,,
\end{aligned}$$

where the third step follows from Lemma A.0.2 and the last step follows from the definition of τ. ■

Lemma A.0.2 *Let τ be an assignment, let X be a configuration such that $X\subseteq\tau(U)$, let x be a point in $\tau(U)$, and let y be a point in $\tau^{-1}(x)$. Then $d(x,X)\le d(y,\tau(y))+d(y,X)$.*

Proof: Let z be a point in X such that $d(y,X)=d(y,z)$. Observe that $d(x,X)\le d(x,z)\le d(x,y)+d(y,z)=d(y,\tau(y))+d(y,X)$. ■

Bibliography

[1] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley, New York, NY, 1991.

[2] S. Arora and R. Kannan. Learning mixtures of arbitrary Gaussians. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 247–257, July 2001.

[3] V. Arya, N. Garg, R. Kandhekar, V. Pandit, A. Meyerson, and K. Munagala. Local search heuristics for k-median and facility location problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 21–29, July 2001.

[4] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, Cambridge, UK, 1998.

[5] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, October 1999.

[6] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 1–10, May 1999.

[7] F. A. Chudak. Improved approximation algorithms for uncapacitated facility location. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes in Computer Science*, pages 180–194, Berlin, 1998. Springer.

[8] S. Dasgupta. Learning mixtures of Gaussians. In *Proceedings of the 40th Annual IEEE Symposium on the Theory of Computation*, pages 634–644, May 1999.

[9] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

[10] B. Everitt. *Cluster Analysis.* Halsted Press, New York, NY, 1980.

[11] J. Frank. *Three-Dimensional Electron Microscopy of MacroMolecular Assemblies.* Academic Press, San Diego, CA, 1996.

[12] J. Frank and Wagenknecht. Automatic selection of molecular images from electron images. *Ultramicroscopy*, 12:169–176, 1984.

[13] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA, 1995.

[14] S. Guha. *Approximation Algorithms for Facility Location Problems.* PhD thesis, Department of Computer Science, Stanford University, August 2000.

[15] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248, 1999.

[16] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data

streams. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 359–366, November 2000.

[17] G. Harauz and A. Fonf-Lochovsky. Automatic particle picking from electron micrographs. *Ultramicroscopy*, 58:333–344, 1989.

[18] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA, 1995.

[19] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 428–434, May 1999.

[20] S. Irani and A. R. Karlin. Online computation. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA, 1995.

[21] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th ACM Symposium on Theory of Computation*, pages 731–740, May 2002.

[22] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48:274–296, 2001.

[23] L. Kavraki, J-C. Latombe, and P. Raghavan. Randomized query processing in robot path planning. *Journal of Computer and System Sciences*, 57:50–60, 1998.

[24] M. Kearns, Y. Mansour, and A. Ng. An information-theoretic analysis of hard and soft assignment methods of clustering. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 282–293, August 1997.

[25] M. Korupolu and M. Dahlin. Coordinated placement and replacement for large-scale distributed caches. In *Proceedings of the IEEE Workshop on Internet Applications*, pages 62–71, July 1999.

[26] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[27] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. *Journal of Algorithms*, 38:260–302, 2001.

[28] A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9:643–666, 1963.

[29] K. R. Lata, P. Renczek, and J. Frank. Automatic particle picking from electronic micrographs. *Ultramicroscopy*, 58:381–391, 1995.

[30] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44:245–249, 1992.

[31] J.-H. Lin and J. S. Vitter. ε-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, May 1992.

[32] B. Lindsay. *Mixture Models: Theory, Geometry, and Applications.* Institute for Mathematical Statistics, Hayward, California, 1995.

[33] R. F. Love, J. G. Morris, and G. O. Wesolowsky. *Facilities Location: Models and Methods.* North-Holland, New York, NY, 1988.

[34] P. D. Mackenzie. Lower bounds for randomized exclusive write PRAMs. In *Proceeding of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 254–263, July 1995.

[35] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[36] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, 1999.

[37] I. M. B. Martin, D. C. Martinescu, R. E. Martin, and T. S. Baker. Identification of spherical virus particles in digitized images of entire electron micrographs. *Journal of Structural Biology*, 120:146–157, 1997.

[38] P. Mirchandani and R. Francis, editors. *Discrete Location Theory.* Wiley, New York, NY, 1990.

[39] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 439–447, January 2001.

[40] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, Cambridge, UK, 1995.

[41] A. Saad, W. Chiu, and P. Thumann-Commike. Multiresolution approach to automatic detection of spherical particles from electron cryomicroscopy images. In *Proceedings of the IEEE 5th Conference on Image Processing*, pages W8–10, 1998.

[42] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, May 1997.

[43] M. Thorup. Quick k-median, k-center, and facility location for sparse graphs. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming*, pages 249–260, July 2001.

[44] P. Thumann-Commike and W. Chiu. Automatic detection of spherical particles from spot-scan electron cryomicroscopy images. *Journal of the Microscopy Society of America*, 1:191–201, 1995.

[45] M. van Heel. Detection of objects in quantum-noise-limited images. *Ultramicroscopy*, 8:331–342, 1982.

[46] A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[47] N. E. Young. K-medians, facility location, and the Chernoff-Wald bound. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 86–95, January 2000.

[48] Z. Yu and C. Bajaj. Particle boxing by natural clustering model. Manuscript, 2002.

Vita

Ramgopal Reddy Mettu was born on November 16, 1975 in Bangalore, Karnataka, India to Sambi Reddy and Samrajyam Mettu; he is the second of three children. He emigrated to the United States in 1982. Ramgopal attended Clear Lake High School in Houston, Texas from 1989 to 1993, where he was chosen as a National Merit Scholar. He enrolled at the University of Texas at Austin in August 1993, where he was a Dedman Merit Scholar from 1993 to 1997. He received a B.S. in Computer Science with Departmental Honors in May 1997; his undergraduate honors thesis project was supervised by Professor Vijaya Ramachandran. He began the Ph.D. program in the Computer Science Department at UT-Austin in August 1997. He received an M.S. in Computer Science in May 1999.

Permanent Address: 14522 Redbud Valley
Houston, TX 77062
U.S.A.