Optimal Time Bounds for Approximate Clustering[†]

Ramgopal R. Mettu
Department of Computer Science
Dartmouth College
Hanover, NH 03755

C. Greg Plaxton
Department of Computer Science
University of Texas at Austin
Austin, TX 78712

Abstract. Clustering is a fundamental problem in unsupervised learning, and has been studied widely both as a problem of learning mixture models and as an optimization problem. In this paper, we study clustering with respect to the k-*median* objective function, a natural formulation of clustering in which we attempt to minimize the average distance to cluster centers. One of the main contributions of this paper is a simple but powerful sampling technique that we call *successive sampling* that could be of independent interest. We show that our sampling procedure can rapidly identify a small set of points (of size just $O(k \log \frac{n}{k})$) that summarize the input points for the purpose of clustering. Using successive sampling, we develop an algorithm for the k-median problem that runs in $O(nk)$ time for a wide range of values of k and is guaranteed, with high probability, to return a solution with cost at most a constant factor times optimal. We also establish a lower bound of $\Omega(nk)$ on any randomized constant-factor approximation algorithm for the k-median problem that succeeds with even a negligible (say $\frac{1}{100}$) probability. The best previous upper bound for the problem was $\tilde{O}(nk)$, where the $\tilde{O}$-notation hides polylogarithmic factors in n and k. The best previous lower bound of $\Omega(nk)$ applied only to deterministic k-median algorithms. While we focus our presentation on the k-median objective, all our upper bounds are valid for the k-means objective as well. In this context our algorithm compares favorably to the widely used k-means heuristic, which requires $O(nk)$ time for just one iteration and provides no useful approximation guarantees.

Keywords: k-means, k-median, unsupervised clustering, approximation algorithms, discrete location theory

[†] This material is based upon work supported by the National Science Foundation under Grants CCR–9821053 and CCR–0310970. The first author was supported by grants to Bruce Donald while at Dartmouth College: from the National Institutes of Health (R01 GM–65982), and the NSF (IIS–9906790, EIA–0102710, EIA–0102712, EIA–9818299, and EIA–9802068, EIA–0305444). Email: `ramgopal@cs.dartmouth.edu`, `plaxton@cs.utexas.edu`. A preliminary version of this work appears in *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence*, Edmonton, Canada, August 2002.

1. Introduction

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 (AK01; CG99; CGTS99; Das99; DH73; Lin95). 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. In this paper, we study the *k-median problem* and 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.

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 (AK01; Das99) 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 $O(k \log \frac{n}{k})$, 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 a running time of $O(nk)$ for k between $\log n$ and $n/\log^2 n$ and, with high probability, produces a solution with cost within a constant factor of optimal.

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* (MS99).) We study a 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.

It is interesting to note that algorithms for the k-median problem can be used for a certain model-based clustering problem as well. The recent work of Arora and Kannan (AK01) formulates 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 solution to this learning problem is to reduce it to the k-median problem and apply an existing constant-factor approximation algorithm for k-median. Thus, our techniques may also have applicability in model-based clustering.

In this paper, we restrict our attention to the ***metric*** version of the k-median problem, in which the n input points are assumed to be drawn from a metric space. That is, the interpoint distances are nonnegative, symmetric, satisfy the triangle inequality, and the distance between points x and y is zero if and only if $x = y$. For the sake of brevity, we write "k-median problem" to mean "metric k-median problem" throughout the remainder of the paper. It is well-known that the k-median problem is NP-hard; furthermore, it is known to be NP-hard to achieve an approximation ratio better than $1 + \frac{2}{e}$ (JMS02). Thus, we focus our attention on developing a k-median algorithm that produces a solution with cost within a constant factor of optimal.

In light of the practical importance of clustering 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.

1.1. Comparison to $\boldsymbol{k}$-means

Even before the hardness results mentioned above were established, heuristic approaches to clustering such as the k-means heuristic were well-studied (see, e.g., (DH73; Mac67; MS99)). 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 (DH73); 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 $2C$.

The k-means heuristic typically uses an objective function that sums 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.

1.2. Our Results

Before stating our results we introduce some useful terminology that we use throughout this paper. Let U denote the set of all points in a given instance of the k-median problem; we assume that U is nonempty. A ***configuration*** is a nonempty subset of U. 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, let $d(x, y)$ denote the distance between x and y, and let $d(x, X)$ be defined as $\min_{y \in X} d(x, y)$. The ***cost*** of any configuration X, denoted $cost\,(X)$, is defined as $\sum_{x \in U} d(x, X) \cdot w(x)$. We denote the minimum cost of any m-configuration by OPT_m. For brevity, we say that an m-configuration with cost at most $a \cdot OPT_k$ is an ***(m, a)-configuration***. 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 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 $r_w = 1 + \lfloor \log R_w \rfloor$.

Our main result is a randomized $O(1)$-approximate k-median algorithm that runs in

$$O\left(\left(n + r_d r_w \log \frac{n}{kr_w}\right) \max\{k, \log n\} + (kr_w)^2\right) \tag{1}$$

time. Note that if $k = \Omega(\log n)$, $kr_w^2 = O(n)$, and $r_d r_w \log \frac{n}{kr_w} = O(n)$, 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 constant factor approximation algorithm for the k-median problem 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 $(k \log \frac{n}{k}$, $O(1))$-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.* (GMMO00) 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 of Mettu and Plaxton (MP03).

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 $(k \log \frac{n}{k}, O(1))$-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.

1.3. Problem Definitions

Without loss of generality, throughout this paper we consider a fixed set of n points, U, with an associated distance function $d : U \times U \rightarrow \mathbb{R}$ and an associated nonnegative demand function $w : U \rightarrow \mathbb{R}$. We assume that d is a metric, that is, d is nonnegative, symmetric, satisfies the triangle inequality, and $d(x, y) = 0$ iff $x = y$. For a configuration X and a set of points Y, we let $\mathit{cost}\,(X, Y) = \sum_{x \in Y} d(x, X) \cdot w(x)$ and we let $\mathit{cost}\,(X) = \mathit{cost}\,(X, U)$. For any set of points X, we let $w(X)$ denote $\sum_{x \in X} w(x)$.

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 $c\,(\tau, X) = \sum_{x \in X} d(x, \tau(x)) \cdot w(x)$.

The input to the k-median problem is (U, d, w) and an integer k, $0 < k \leq n$. Since our goal is to obtain a $(k, O(1))$-configuration, we can assume without loss of generality that all input points have nonzero weight. We note that for all m, $0 < m \leq n$, removing zero weight points from an m-configuration at most doubles its cost. To see this, consider an m-configuration X; we can obtain an m-configuration X' by replacing each zero weight point with its closest nonzero weight point. Using the triangle inequality, it is straightforward to see that $\mathit{cost}\,(X') \leq 2\mathit{cost}\,(X)$. This argument can be used to show that any minimum-cost set of size m contained in the set of nonzero weight input points has cost at most twice OPT_m. We also assume that the input weights are scaled such that the smallest weight is 1; thus the input weights lie in the range $[1, R_w]$. For output, the k-median problem requires us to compute a minimum-cost k-configuration. The ***uniform weights*** k-median problem is the special case in which $w(x)$ is a fixed real for all points x. The output is also a minimum-cost k-configuration.

1.4. Previous Work

The first $O(1)$-approximate k-median algorithm was given by Charikar *et al.* (CGTS99). Subsequently, there have been several improvements to the approximation ratio (see, e.g., (CG99) for results and citations). In this section, we focus on the results that are most relevant to the present paper; we compare our results with other recent randomized algorithms for the k-median problem. The first of these results is due to Indyk, who gives a randomized $(O(k), O(1))$-approximate algorithm for the uniform weights k-median problem (Ind99) that runs in $\tilde{O}(nk/\delta^2)$ time, where δ is the desired failure probability.

Thorup (Tho01) gives randomized $O(1)$-approximate algorithms for the k-median, k-center, and facility location problems in a graph. For these problems, 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 the algorithms in (Tho01) run in $\tilde{O}(m)$ time. Thorup (Tho01) 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 but produces an $(O((k \log^2 n)/\varepsilon), 2+\varepsilon)$-configuration for any positive real ε with $0 < \varepsilon < 0.4$, and is only guaranteed to succeed with probability $1/2$.

For the data stream model of computation, Guha *et al.* (GMMO00) 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. They also establish a lower bound of $\Omega(nk)$ for deterministic $O(1)$-approximate k-median algorithms.

Mishra *et al.* (MOP01) 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 $o(n^2/k)$).

1.5. Outline

The rest of this paper is organized as follows. In Sections 2 and 3, we present and analyze our successive sampling algorithm. In Section 4, we make use of our sampling algorithm, in conjunction with an extraction result, to develop an $O(1)$-approximate uniform weights k-median algorithm. Then, in Section 5, we use the uniform weights algorithm as a subroutine to develop an $O(1)$-approximate k-median algorithm for the case of arbitrary weights. We present our lower bounds for the k-median problem and its clustering variant in Section 6. We offer some concluding remarks in Section 7.

2. Approximate Clustering via 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 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 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 union of all balls (x, r) such that $x \in X$.

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 $O(\log \frac{w(U)}{k'})$. For the uniform demands k-median problem, t is simply $O(\log \frac{n}{k'})$. 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 fourth and fifth steps are negligible. Thus, for the uniform demands k-median problem, the total running time of the above algorithm is $O(nk')$.

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 1 below; this result is used to analyze the algorithms of Sections 4 and 5. The proof of the theorem makes use of Lemma 3.3, established in Section 3.1, and Lemmas 3.5 and 3.11, established in Section 3.2.

THEOREM 1. *With high probability, $c(\sigma) = O(cost(X))$ for any k-configuration X.*

Proof: The claim of Lemma 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 3.3, 3.5, and 3.11. $\square$

Before proceeding, we give some intuition behind the proof of Theorem 1. The proof consists of two main parts. First, Lemma 3.3 shows that with high probability, for i such that $0 \leq i \leq t$, 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 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 1 follows immediately from Lemma 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 1 we provide a more careful accounting of the cost of an optimal k-configuration. Specifically, in

Section 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 3.11 formalizes this statement and proves a lower bound on the cost of an optimal k-configuration, and Lemma 3.5 completes the proof of Theorem 1 by providing an upper bound on the cost of σ.

3.1. Balls and Bins Analysis

The proof of Lemma 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., (MR95)) can be used to obtain sharp estimates on $f(m, a, b)$. However, the following simple upper bound is sufficient for our purposes.

LEMMA 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: Imagine throwing the $\lceil \lambda b \rceil$ balls one at a time into the m bins, and consider a throw to be *successful* if it lands in a previously empty bin. Thus $f(m, \varepsilon, \lambda b)$ is the probability that the number of successful throws is less than $(1 - \varepsilon)m$. While the number of successful throws is less than $(1-\varepsilon)m$, each successive throw has success probability greater than ε. It follows that an upper bound on $f(m, \varepsilon, \lambda b)$ is given by the probability of obtaining at most $(1-\varepsilon)m$ successes in $\lceil \lambda b \rceil$ independent Bernoulli trials, each of which has success probability ε. The claim then follows by choosing λ sufficiently large and applying the tail bound on the binomial distribution given by Equation (4) in Appendix A. $\square$

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 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 2, let $v = (r_0, \ldots, r_{m-1})$ and let R denote the sum of the r_i's.

We will use Lemma 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 Equation (4) in Appendix A, 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 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 empty complete sub-bins is at most $s \cdot \frac{\varepsilon}{2} \cdot \frac{2m}{\varepsilon} = \varepsilon R/2$. 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 the minimum 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 3.3 below establishes the main probabilistic claim used in our analysis of the algorithm of Section 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 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, Equation (4) in Appendix A 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 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. □

3.2. Upper and Lower Bounds on Cost

Recall that Lemma 3.3 alone is not sufficient to establish Theorem 1 (see the text preceding Section 3.1 for an informal discussion). In this section, we give the analysis needed to complete the proof of Theorem 1 by providing an upper bound on the cost of the assignment σ as well a lower bound on the cost of an optimal k-configuration. Lemmas 3.4 and 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 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 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 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 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 3.7. *Let i be an integer such that $0 \le i \le t$ and let Y be a subset of F_i. Then $w(F_i) \ge (1-\gamma)w(U_i)$ and $\mathit{cost}\,(X,Y) \ge \mu_i w(Y)$.*

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, $d(y,X) \ge \mu_i$ for any y in F_i. Thus $\mathit{cost}\,(X,Y) = \sum_{y \in Y} d(y,X) \cdot w(y) \ge \mu_i w(Y)$. □

LEMMA 3.8. *For all integers ℓ and m such that $0 \le \ell \le t$ and $m > 0$,*

$$\mathit{cost}\left(X, \cup_{i \in G_{\ell,m}} F_i^m\right) \ \ge \sum_{i \in G_{\ell,m}} \mu_i w(F_i^m).$$

Proof: By Lemma 3.6, for all ℓ and m such that $0 \le \ell \le t$ and $m > 0$,

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

By Lemma 3.7, $\mathit{cost}\,(X, F_i^m) \ge \mu_i w(F_i^m)$, and the claim follows. □

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

LEMMA 3.9. *For all i such that $0 \le i \le t$, $w(F_{i+r}) \le \frac{1}{3} w(F_i)$.*

Proof: Note that $w(F_{i+r}) \le w(U_{i+r}) \le (1-\beta)^r w(U_i) \le \frac{(1-\beta)^r}{1-\gamma} w(F_i)$, where the last step follows from Lemma 3.7. The claim then follows by the definition of r. □

LEMMA 3.10. *For all i such that $0 \le i \le t$, $w(F_i^r) \ge \frac{w(F_i)}{2}$.*

Proof: Observe that

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

where the second step follows from Lemma 3.9. □

LEMMA 3.11. *For any k-configuration X,*

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

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

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

where the first step follows from Lemma 3.8, the second step follows from averaging and the choice of ℓ, the third step follows from Lemma 3.10, the fourth step follows from Lemma 3.7, and the last step follows since $C_i \subseteq U_i$. □

4. An Efficient Algorithm for the Case of Uniform Weights

In this section we obtain a fast k-median algorithm for the case of uniform weights. Our algorithm makes use of the sampling algorithm of Section 2, a black-box k-median algorithm and a slight generalization of algorithm Small-Space of Guha *et al.* (GMMO00) that we call Modified-Small-Space. Section 4.1 below gives the analysis required to generalize algorithm Small-Space of Guha *et al.* In Section 4.2 we describe our uniform weights algorithm in detail, including a discussion of the approximation bound and running time.

4.1. Algorithm Modified-Small-Space

In this subsection we establish that a modified version of algorithm Small-Space of Guha *et al.* (GMMO00) 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 and 5. We note that the changes to the algorithm of Guha *et al.* are trivial; the discussion in this section is included for completeness.

We now discuss the modification to algorithm Small-Space of Guha *et al.* and the changes required in the analysis. In Step 2 of algorithm Small-Space, 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.* (GMMO00) 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 (GMMO00, Theorem 2.3) (which is used in the proof of (GMMO00, 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.* for algorithm Modified-Small-Space as Theorem 2 below and give the required generalization of Theorem 2.3 with Lemma 4.1 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 $c_\tau(X)$ denote $\sum_{x \in \tau(U)} d(x, X) \cdot w_\tau(x)$.

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 2 (Guha *et al.* (GMMO00)). *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 4.1. *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,*

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

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 4.2 and the last step follows from the definition of τ. □

LEMMA 4.2. *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 $d(x,X)\le d(x,z)\le d(x,y)+d(y,z)=d(y,\tau(y))+d(y,X)$. □

4.2. Algorithm

We obtain our uniform weights k-median algorithm by applying our sampling algorithm in Step 2 of algorithm Modified-Small-Space and the deterministic online median algorithm of Mettu and Plaxton (MP03) in Step 4. 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 1, the output of our sampling algorithm is an $(m,\ O(1))$-assignment with high probability, where $m=O(\max\{k,\log n\}\log\frac{n}{k})$. The online median algorithm of Mettu and Plaxton (MP03) is also an $O(1)$-approximate k-median algorithm. Thus, by Theorem 2, 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 Mettu and

Plaxton (MP03) requires $O(|\sigma(U)|^2 + |\sigma(U)|\, r_d)$ time. The total time taken by the algorithm is therefore

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

where the first step follows from the analysis of our sampling algorithm for the case of uniform weights. The second step follows from the fact that $k' \log^2 \frac{n}{k}$ is $O(n)$. By the choice of k', the overall running time is $O((n + r_d \log \frac{n}{k}) \max\{k, \log n\})$. Note that if $k = \Omega(\log n)$ and $r_d \log \frac{n}{k} = O(n)$, this time bound simplifies to $O(nk)$.

5. An Efficient Algorithm for the Case of Arbitrary Weights

The algorithm developed in Sections 2 and 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 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 2 and 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 2 and 4, and let $\mathcal{B}$ be an $O(1)$-approximate k-median algorithm.

- Compute sets B_i for $0 \le i < r_w$ such that for all $x \in B_i$, $2^i \le w(x) \le 2^{i+1}$.
- 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 1 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 is used in step 2 of Small-Space, and the online median algorithm of Mettu and Plaxton (MP03) is used in step 4 of Small-Space. Thus, Theorem 2 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

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

(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 (MP03) 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 (1).

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 (Yao77) due to Mackenzie (Mac97), we can establish an upper 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 deterministic 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 $f(S)$ as follows: For all positive reals ε and c, there exist positive reals δ and γ and a positive integer n_0 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 later in the section we will often be interested in statements of the form $f(S)$ for some S that has no explicit dependence on one or more of the quantified variables ε, c, δ, γ, n_0, n, and k introduced above. We also remark that if $f(S)$ and $f(T)$ hold, then $f(S \wedge T)$ holds.)

We define $f'(S)$ in the same way as $f(S)$ except that the restriction on k is strengthened to $1 < k < \frac{3n}{4}$. Similarly, $f''(S)$ is the same as $f(S)$ except that the restriction on k is $\frac{3n}{4} \leq k < n$. Note that for any statement S, $f'(S) \wedge f''(S)$ implies $f(S)$.

Finally, for addressing the k-median problem we define statements $g(S)$, $g'(S)$, and $g''(S)$ in 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 statement $f($"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 statement $g($"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 6.1 and 6.2 below. The proof of the second theorem follows from Lemmas 6.3 and 6.4.

LEMMA 6.1. *The statement $f'($"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: 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 $n - k$.

In the following definitions, we make use of a parameter δ' that will be chosen sufficiently small with respect to ε and sufficiently large with respect to δ. We call a point x *heavy* if $\mathcal{A}$ probes the distance between x and more than $\delta' k$ other points. We call a point x *lucky* if one of $\mathcal{A}$'s first $\delta' k$ probes involving x is to a point y such that $d(x, y) = 1$. We call a point x *clean* with respect to an execution of algorithm $\mathcal{A}$ if it is neither heavy nor lucky.

We first observe that the statement $S_1 = f'($"at least $(1-\varepsilon)n$ points are not heavy"$)$ holds since $\mathcal{A}$ is allowed at most δnk probes and δ can be made arbitrarily small with respect to δ'.

Let us now choose an arbitrary point x and derive an upper bound on the probability that x is lucky. Assume without loss of generality that, before $\mathcal{A}$ makes any probes involving x, it knows the distance between all pairs of points not containing x. (Knowing this information could only enhance the probability that x is lucky.) Under this assumption, the probability that x is lucky is at most $\delta' k/k = \delta'$ since x is equally likely to belong to any of the k equivalence classes. By linearity of expectation, the expected number of lucky points is at most $\delta' n$. By Markov's inequality, the probability that the number of lucky points exceeds m times the expected number is at most $1/m$ for all $m > 0$. Hence the statement $S_2 = f'($"at least $(1 - \varepsilon)n$ of the points are not lucky"$)$ holds.

By combining statements S_1 and S_2, we have $S_3 = f'($"at least $(1 - \varepsilon)n$ points are clean"$)$. (Remark: The reader might believe that the factor of $(1 - \varepsilon)$ appearing in statement S_3 should be $(1 - 2\varepsilon)$ to account for the two factors of $(1 - \varepsilon)$ appearing in statements S_1 and S_2. Note, however, that the parameter ε is universally quantified in each of these three statements. Let $T_1(\varepsilon)$ (resp., $T_2(\varepsilon)$, $T_3(\varepsilon)$) denote the statement S_1 (resp., S_2, S_3) with no universal quantification over ε. Note that $T_3(\varepsilon)$ follows from $T_1(\varepsilon/2)$ and $T_2(\varepsilon/2)$. Thus S_3 follows from S_1 and S_2.)

Since $\mathcal{A}$ is a k-matching algorithm, it outputs at least $\frac{n-k}{2} \geq n/8$ pairs. This observation, together with statement S_3 above, implies $f'($"at least $n/9$ of the pairs produced by $\mathcal{A}$ consist of two clean points"$)$. Each such output pair of clean points independently contributes ℓ to the cost of the k-matching produced by $\mathcal{A}$ with probability at least $1 - \frac{1}{k(1-\delta')}$. The claim of the lemma now follows by Equation (4) in Appendix A, assuming that we choose δ sufficiently small, and γ and n_0 sufficiently large, as a function of ε and c. □

LEMMA 6.2. *The statement f''("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: The proof of the preceding lemma does not readily extend to large values of k, so we resort to 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 6.1, the cost of an optimal k-clustering solution is $n-k$.

In the following, let a denote an integer parameter to be chosen sufficiently large with respect to $1/\varepsilon$. 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) independently place each of the n points uniformly at random into one of $\lfloor \frac{n}{a} \rfloor$ *supergroups*; (2) randomly partition each supergroup of size s into $\lfloor s/2 \rfloor$ pairs and, if s is odd, one singleton; (3) split all but a random subset of $n-k$ of the pairs into singletons. (Note that the number of pairs formed in step (2) is at least $\frac{n-\lfloor n/a \rfloor}{2}$. Since a is to be chosen sufficiently large, we can assume that $a \geq 2$ and hence this quantity is at least $n/4$. Since $k \geq 3n/4$, the number of pairs formed in step (2) is at least $n-k$; it follows that step (3) is well-defined.) In what follows we refer to the resulting $n-k$ pairs and $2k-n$ 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*. In addition, we refer to the pairs identified in step (2) as *intermediate-pairs*.

We define a supergroup to be *good* if it contains between $a/2$ and $2a$ points. Equations (4) and (2) in Appendix A can be used to upper bound the (small) probability that an individual supergroup is bad (i.e., not good). This upper bound, together with Markov's inequality, yields f''("at least a $1-\varepsilon$ fraction of the supergroups are good"), which we refer to below as statement S_1. Furthermore, Equation (3) in Appendix A can be used to establish $S_2 = f''$("at least a $1-\varepsilon$ fraction of the points belong to good supergroups").

We define a point x to be *clean* with respect to an execution of $\mathcal{A}$ if $\mathcal{A}$ never probes the distance between x and another point in the same supergroup as x. We define a supergroup G to be *clean* if every point in G is clean. By an argument similar to that used to establish f'("at least a $1-\varepsilon$ fraction of the points are clean") in the proof of Lemma 6.1, we can show that with probability at least $1-\delta'$, at least a $1-\delta'$ fraction of the points are clean, where δ' can be made arbitrarily small compared to $1/a$ (which in turn is small compared to ε) by choosing δ sufficiently small compared to $1/a$. Observing that the number of dirty (i.e., not clean) supergroups is at most the number of dirty points, we obtain $S_3 = f''$(“at least a $1-\varepsilon$ fraction of the supergroups are clean”).

Combining statements S_1, S_2, and S_3, we obtain $S_4 = f''$(“at least a $1-\varepsilon$ fraction of the points belong to clean good supergroups”). Statement S_4 implies $S_5 = f''$(“at least a $1-\varepsilon$ fraction of the intermediate-pairs belong to clean good supergroups”). Thus in what follows we may assume without loss of generality that at most an ε fraction of the intermediate-pairs belong to bad or dirty supergroups. Let A denote the set of all such intermediate-pairs. Augment A as necessary with other intermediate-pairs to obtain a set B with exactly $\lfloor \varepsilon(n-k) \rfloor$ intermediate-pairs. By the principle of deferred decisions, in particular, by deferring the random choice of the set of $n-k$ input-pairs from the set of intermediate-pairs, the expected fraction of the input-pairs contained in B is at most ε. Since B contains A, the same inequality holds for A. Using Markov's inequality, we now obtain $S_6 = f''$(“at least a $1-\varepsilon$ fraction of the input-pairs belong to clean good supergroups”).

Let us define a supergroup to be *interesting* if it is clean, good, and contains at least one input-pair. Since there are $n-k$ input-pairs and a good supergroup cannot contain more than a input-pairs, statement S_6 implies $S_7 = f''$(“there are at least $\frac{n-k}{2a}$ 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 interesting 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 establishes the lemma by statement S_7, assuming a sufficiently large choice for the parameter $\gamma = \ell$. Thus, in what follows, we may assume that at least half of the interesting supergroups are red.

Let G denote an interesting red supergroup and let (x, y) denote an output-pair belonging to G; such a pair exists since G is red. If x is an input-singleton then the cost of output-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 $y = z$. Because G is interesting, it is clean and good. Hence G contains at least $a/2$ points and, using the principle of deferred decisions, the probability that $y = z$ is at most $\frac{1}{(a/2)-1}$. (Here we are deferring the choice of the intermediate-pairs within a clean supergroup.) By choosing a sufficiently large compared to $1/\varepsilon$, we can ensure that the latter probability is arbitrarily smaller than ε. Furthermore, the event that $y = z$ is independent of the analogous events defined for other interesting red supergroups. Thus each interesting red supergroup independently contributes, with probability $1-p$ where p is arbitrarily small compared to ε, 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 Equation (4) in Appendix A, with the parameters a, δ, γ, and n_0 set to suitably chosen functions of ε and c. □

LEMMA 6.3. *The statement g'("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: 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 in the same tentative equivalence class as x for which $\mathcal{A}$ has probed $d(x, y)$.

It is not difficult to establish g'("at least $(1-\varepsilon)n$ points are clean"). The proof of this statement is omitted since it is similar to the proof of the statement f'("at least $(1-\varepsilon)n$ points are clean") that is given within the proof Lemma 6.1. Since the special points are chosen uniformly at random, it follows that g'("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 g'("$\mathcal{A}$ fails to output $\frac{1}{8}$ (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 g'("the cost of the solution computed by $\mathcal{A}$ is at least $(1-\varepsilon)k\ell/16$"). Choosing γ sufficiently large (depending on c), the claim of the lemma then follows since $\ell = \gamma n/k$. □

LEMMA 6.4. *The statement g''("the cost of the k-median solution computed by $\mathcal{A}$ is more than c times optimal") holds.*

Proof: This proof is similar to that of Lemma 6.2 above. We define the input distribution D in the same manner, as well as the following terms: supergroup, good supergroup, clean supergroup, interesting supergroup, input-pair, input-singleton. We note that since these definitions refer only point sets that are drawn from the input distribution D, they remain valid even though we now assume $\mathcal{A}$ to be a k-median algorithm. Furthermore, we can use arguments similar to those used to establish statement S_7 in Lemma 6.2 to establish the statement $T = g''$("at least $\frac{n-k}{2a}$ of the supergroups are interesting"), where a is an integer parameter to be chosen sufficiently large with respect to $1/\varepsilon$.

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 each 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 establishes the lemma by statement T above. 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).

Let G denote an 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 $i \geq 1$ (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 $\binom{b}{i}$, where $b = 2i + j$ denotes the cardinality of G. Given that $i \geq 1$ and $j \geq 0$, this probability is at most $2/b$. Since G is interesting (and hence good), we have $a/2 \leq b$ and so $2/b \leq 4/a$. Furthermore, the event that $\mathcal{A}$ produces such an output is independent of the analogous events defined for other interesting balanced supergroups. Thus each interesting balanced supergroup independently contributes, with probability at least $1 - 4/a$, 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 Equation (4) in Appendix A, with the parameters a, δ, γ, and n_0 set to suitably chosen functions of ε and c. □

7. Concluding Remarks

In this paper, we have presented a constant-factor approximation algorithm for the k-median problem that runs in optimal $\Theta(nk)$ time if $\log n \leq k \leq \frac{n}{\log^2 n}$. If we use our algorithm as an initialization procedure for k-means, our analysis guarantees that the cost of the output of k-means is within a constant factor of optimal. Preliminary experimental work (Met02) suggests that this approach to clustering yields improved practical performance in terms of running time and solution quality.

References

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.

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

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.

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.

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

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

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.

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.

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.

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

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.

P. D. MacKenzie. Lower bounds for randomized exclusive write PRAMs. *Theory of Computing Systems*, 30:599–626, 1997.

R. R. Mettu. *Approximation Algorithms for NP-Hard Clustering Problems.* PhD thesis, Department of Computer Science, University of Texas at Austin, August 2002.

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.

R. R. Mettu and C. G. Plaxton. The online median problem. *SIAM Journal on Computing*, 32:816–832, 2003.

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

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

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.

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.

Appendix

A. Tail Bounds for the Binomial Distribution

In this section we state several standard bounds on the tail of the binomial distribution. See, for example, the text by Alon and Spencer (AS91, Appendix A) for derivations of these inequalities.

Let n be a nonnegative integer and let p be a real $[0, 1]$. Let X denote the random variable corresponding to the total number of successes in n independent Bernoulli trials, each of which succeeds with probability p. The random variable X is said to be *binomially distributed with parameters n and p.* Note that $E(X) = np$; let μ denote $E(X)$.

The following pair of inequalities are useful for bounding the upper tail of the binomial distribution. The first is valid for all δ in $[0, 1]$:

$$\Pr(X \geq (1+\delta)\mu) \ \leq \ e^{-\delta^2\mu/3} \tag{2}$$

The second holds for all $\delta \geq 0$:

$$\Pr(X \geq (1+\delta)\mu) \ \leq \ \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \tag{3}$$

The following inequality is useful for bounding the lower tail of the binomial distribution; it is valid for all δ in $[0, 1]$.

$$\Pr(X \leq (1-\delta)\mu) \ \leq \ e^{-\delta^2\mu/2} \tag{4}$$