How Fast is the k-means Method?[*]

Sariel Har-Peled[†] Bardia Sadri[‡]

January 2, 2010

Abstract

We present polynomial upper and lower bounds on the number of iterations performed by the k-means method (a.k.a. Lloyd's method) for k-means clustering. Our upper bounds are polynomial in the number of points, number of clusters, and the spread of the point set. We also present a lower bound, showing that in the worst case the k-means heuristic needs to perform $\Omega(n)$ iterations, for n points on the real line and two centers. Surprisingly, the spread of the point set in this construction is *polynomial*. This is the first construction showing that the k-means heuristic requires more than a polylogarithmic number of iterations. Furthermore, we present two alternative algorithms, with guaranteed performance, which are simple variants of the k-means method. Results of our experimental studies on these algorithms are also presented.

1 Introduction

In a (geometric) *clustering* problem, we are given a finite set $X \subset \mathbb{R}^d$ of n points and an integer $k \geq 2$, and we seek a partition (clustering) $\mathcal{S} = (S_1, \ldots, S_k)$ of X into k disjoint nonempty subsets along with a set $C = \{c_1, \ldots, c_k\}$ of k corresponding *centers*, that minimizes a suitable cost function among all such k-clusterings of X. The cost function typically represents how tightly each cluster is packed and how separated different clusters are. A center c_i *serves* the points in its cluster S_i.

We consider the *k-means* clustering *cost function* $\phi(\mathcal{S}, C) = \sum_{i=1}^{k} \psi(S_i, c_i)$, in which $\psi(S, c) = \sum_{x \in S} \|x - c\|^2$, where $\|\cdot\|$ denotes the Euclidean norm. It can be easily observed that for any cluster S_i, the point c that minimizes the sum $\sum_{x \in S_i} \|x - c\|^2$, is the centroid of S_i, denoted by $c(S_i)$, and therefore in an optimal clustering, $c_i = c(S_i)$. Thus the above cost function can be written as $\phi(\mathcal{S}) = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c(S_i)\|^2$.

It can also be observed that in an optimal k-clustering, each point of S_i is closer to c_i, the center corresponding to S_i, than to any other center. Thus, an optimal k-clustering is imposed by a Voronoi diagram whose sites are the centroids of the clusters. Such partitions are related to centroidal Voronoi tessellations (see [DFG99]).

A k-means clustering algorithm that is used widely because of its simplicity is the *k-means heuristic*, also called *Lloyd's method* [Llo82]. This algorithm starts with an arbitrary k-clustering $\mathcal{S}_0$ of X with the initial k centers chosen to be the centroids of the clusters of $\mathcal{S}_0$. Then it repeatedly performs local improvements by applying the following "hill-climbing" step.

[*]The most up-to-date version of this paper is available from the author's web page: http://www.uiuc.edu/~sariel/papers/03/lloyd_kmeans.

[†]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@uiuc.edu; http://www.uiuc.edu/~sariel/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

[‡]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL 61801; USA; http://www.uiuc.edu/~sadri/; sadri@uiuc.edu.

Definition 1.1 Given a clustering $\mathcal{S} = (S_1, \ldots, S_k)$ of X, a k-Means *step* returns a clustering $\mathcal{S}' = (S'_1, \ldots, S'_k)$ by letting S'_i equal to the intersection of X with the cell of $c(S_i)$ in the Voronoi partitioning imposed by centers $c(S_1), \ldots, c(S_k)$. If a point of X has more than a single closest center, it is assigned to one of its incident Voronoi cells arbitrarily. The (new) center of S'_i will be $c(S'_i)$.

In a clustering $\mathcal{S} = (S_1, \ldots, S_k)$ of X, a point $x \in X$ is *misclassified* if there exists $1 \leq i \neq j \leq k$, such that $x \in S_i$ but $\|x - c(S_j)\| < \|x - c(S_i)\|$. Thus a k-Means step can be broken into two stages: (i) every misclassified point is assigned to its closest center (with ties broken arbitrarily), and (ii) all centers are moved to the centroids of their newly formed clusters.

The k-means algorithm, to which we shall refer as "k-MeansMtd" throughout this paper, performs the k-Means step repeatedly and stops when the assignment of the points to the centers does not change from that of the previous step. This happens when there remains no misclassified points and consequently in the last k-Means step $\mathcal{S}' = \mathcal{S}$. Clearly the clustering cost is reduced when each point is mapped to the closest center and also when each center moves to the centroid of the points it serves. Thus, the clustering cost is strictly reduced in each of the two stages of a k-Means step. This in particular implies that no clustering can be seen twice during the course of execution of k-MeansMtd. Since there are only finitely many k-clusterings, the algorithm terminates in finite time.

The algorithm k-MeansMtd and its variants are widely used in practice [DHS01]. Convergence and consistency of k-MeansMtd in probabilistic settings is studied in [Mac67] and [Pol81]. It is known that the output of k-MeansMtd is not necessarily a global minimum, and it can be arbitrarily bad compared to the optimal clustering. Furthermore, the answer returned by the algorithm and the number of steps depend on the initial choice of the centers, i.e. the initial clustering [KMN+02]. These shortcomings of k-MeansMtd has led to development of efficient polynomial approximation schemes for the k-means clustering problem both in low [Mat00, ES03, HM04, KSS04] and vhigh dimensions [dlVKKR03]. Unfortunately, those algorithms have had little impact in practice, as they are complicated and probably impractical because of large constants. A more practical local search algorithm, which guarantees a constant factor approximation, is described by Kanungo *et al.* [KMN+02].

Up to this point, no meaningful theoretical bound was known for the number of steps k-MeansMtd can take to terminate in the worst case. Inaba *et al.* [IKI94] observe that the number of distinct Voronoi partitions of a given n-point set $X \subset \mathbb{R}^d$ induced by k sites is at most $O(n^{kd})$ which gives a trivial similar upper bound on the number of steps of k-MeansMtd (by observing that the clustering cost monotonically decreases and thus no k-clustering can be seen twice). However, the fact that k in typical application can be in the hundreds together with the relatively fast convergence of k-MeansMtd observed in practice, make this bound somewhat meaningless. The difficulty of proving any super-linear lower bound further suggests the looseness of this bound.

Our contribution. It thus appears that the combinatorial behavior of k-MeansMtd is far from being well understood. Motivated by this, in this paper we provide a lower bound and several upper bounds on the number of iterations performed by k-MeansMtd and some of its close variants. To our knowledge, our lower bound is the first that is super-polylogarithmic. Our upper bounds are *polynomial* in the spread Δ of the input point set, k, and n (the *spread* of a point set is the ratio between its diameter and the distance between its closest pair). The bounds are meaningful (i.e., polynomial) for most inputs.

In Section 2, we present an $\Omega(n)$ lower bound on the number of iterations performed by k-MeansMtd. More precisely, we show that for an adversarially chosen initial two centers and a set of n points on the line, k-MEANSMTD takes $\Omega(n)$ steps. Note, that this matches the straightforward upper bound on the number of Voronoi partitions in one dimension with two centers, which is $O(n)$.

In Section 3, we provide a polynomial upper bound for the one-dimensional case. In Section 4, we provide an upper bound for the case where the points lie on a grid. In Section 5, we investigate two alternative algorithms, and provide polynomial upper bounds on the number of iterations they perform. Those algorithms are minor modifications of k-MEANSMTD algorithm, and we believe that their analysis provides an insight about the behavior of k-MEANSMTD. Some experimental results are presented in Section 6. In Section 7, we conclude by mentioning a few open problems and a discussion of our results.

2 Lower Bound Construction for Two Clusters in One Dimension

In this section, we describe a set of $2n$ points, along with an initial pair of centers, on which k-MEANSMTD takes $\Omega(n)$ steps.

Fix $n \geq 2$. Our set X will consist of $2n$ numbers $y_1 < \cdots < y_n < x_n < \cdots < x_1$ with $y_i = -x_i$, for $i = 1, \ldots, n$.

At the ith iteration, we denote by l_i and r_i the current left and right centers, respectively, and by L_i and R_i the new sets of points assigned to l_i and r_i, respectively. Furthermore, for each $i \geq 0$, we denote by α_i the Voronoi boundary $\frac{1}{2}(l_i + r_i)$ between the centers l_i and r_i. Thus

$$L_i = \{x \in X \mid x < \alpha_i\} \qquad \text{and} \qquad R_i = \{x \in X \mid x \geq \alpha_i\}\,.$$

Let x_1 be an arbitrary positive real number and let $x_2 < x_1$ be a positive real number to be specified shortly. Initially, we let $l_1 = x_2$ and $r_1 = x_1$ and consequently $\alpha_1 = \frac{1}{2}(x_1 + x_2)$. Thus in the first iteration, $L_1 = \{y_1, \ldots, y_n, x_n, \ldots, x_2\}$ and $R_1 = \{x_1\}$. We will choose $x_2, \ldots, x_n$ such that at the end of the ith step we have $L_i = \{y_1, \ldots, y_n, x_n, \ldots, x_{i+1}\}$ and $R_i = \{x_i, \ldots, x_1\}$. Suppose for the inductive hypothesis that at the $(i-1)$th step we have

$$L_{i-1} = \{y_1, \ldots, y_n, x_n, \ldots, x_{i+1}, x_i\} \qquad \text{and} \qquad R_{i-1} = \{x_{i-1}, \ldots, x_1\}\,.$$

Thus we can compute l_i and r_i as follows

$$l_i = \frac{y_1 + \cdots + y_n + x_n + \cdots + x_i}{2n - i + 1} \qquad \text{and} \qquad r_i = \frac{x_{i-1} + \cdots + x_1}{i - 1}.$$

Since $y_1 + \cdots + y_n + x_n + \cdots + x_i = -(x_{i-1} + \cdots + x_1)$, we get for α_i:

$$\begin{aligned}
\alpha_i = \frac{1}{2}(l_i + r_i) &= \frac{1}{2}\left(\frac{x_{i-1} + \cdots + x_1}{i-1} - \frac{x_{i-1} + \cdots + x_1}{2n - i + 1}\right) \\
&= \frac{n - i + 1}{(i-1)(2n - i + 1)}(x_{i-1} + \cdots + x_1) = \frac{n - i + 1}{(i-1)(2n - i + 1)} \cdot s_{i-1},
\end{aligned}$$

where $s_{i-1} = \sum_{j=1}^{i-1} x_j$.

To guarantee that only x_i deserts L_{i-1} to R_i, in the ith iteration, we need that $x_{i+1} < \alpha_i < x_i$. Thus, it is natural to set $x_i = \tau_i \alpha_i$, where $\tau_i > 1$, for $i = 1, \ldots, n$. Picking the coefficients $\tau_1, \ldots, \tau_n$ is essentially the only part of this construction that is under our control. We set

$$\tau_i = 1 + \frac{1}{n - i + 1} = \frac{n - i + 2}{n - i + 1},$$

for $i = 1, \ldots, n$. Since $\tau_i > 1$, $x_i = \tau_i \alpha_i > \alpha_i$, for $i = 1, \ldots, n$. Next, we verify that $x_{i+1} < \alpha_i$. By definition,

$$\begin{aligned}
x_{i+1} &= \tau_{i+1}\alpha_{i+1} = \tau_{i+1} \cdot \frac{n-i}{i(2n-i)} \cdot s_i \\
&= \tau_{i+1} \cdot \frac{n-i}{i(2n-i)} \cdot (x_i + s_{i-1}) = \tau_{i+1} \cdot \frac{n-i}{i(2n-i)} \left(\tau_i + \frac{(i-1)(2n-i+1)}{n-i+1} \right) \alpha_i \\
&= \frac{n-i+1}{i(2n-i)} \left(\frac{n-i+2}{n-i+1} + \frac{(i-1)(2n-i+1)}{n-i+1} \right) \alpha_i.
\end{aligned}$$

It can be verified through elementary simplifications that the coefficient of α_i above is always less than 1 implying that $x_{i+1} < \alpha_i < x_i$, for $i = 1, \ldots, n-1$.

We can compute a recursive formula for x_{i+1} in terms of x_i, as follows

$$\begin{aligned}
x_{i+1} &= \tau_{i+1}\alpha_{i+1} = \frac{n-i+1}{n-i} \cdot \frac{n-i}{i(2n-i)} \cdot s_i = \frac{n-i+1}{i(2n-i)} \cdot (x_i + s_{i-1}) \\
&= \frac{n-i+1}{i(2n-i)} \left(x_i + \frac{(i-1)(2n-i+1)}{n-i+1} \cdot \alpha_i \right) \\
&= \frac{n-i+1}{i(2n-i)} \left(x_i + \frac{(i-1)(2n-i+1)}{n-i+1} \left(1 + \frac{1}{n-i+1} \right)^{-1} x_i \right) \\
&= \frac{n-i+1}{i(2n-i)} \left(1 + \frac{(i-1)(2n-i+1)}{n-i+1} \left(\frac{n-i+1}{n-i+2} \right) \right) x_i. \\
&= \frac{n-i+1}{i(2n-i)} \left(1 + \frac{(i-1)(2n-i+1)}{n-i+2} \right) \cdot x_i,
\end{aligned}$$

for $i = 1, \ldots, n-1$. Thus letting $\beta_i = \frac{n-i+1}{i(2n-i)} \left(1 + \frac{(i-1)(2n-i+1)}{n-i+2} \right)$ we get that

$$x_{i+1} = \beta_i x_i, \tag{1}$$

for $i = 1, \ldots, n-1$.

Theorem 2.1 *For each $n \geq 2$, there exists a set of $2n$ points on a line with two initial center positions for which k-MeansMtd takes exactly n steps to terminate.*

2.1 The Spread of the Point Set

It is interesting to examine the spread of the above construction. As we show below, somewhat surprisingly, the spread of this construction is polynomial, hinting (at least intuitively) that "bad" inputs for k-MeansMtd are, arguably, not that contrived.

By Eq. (1), we have $x_{i+1} = \beta_i x_i$. Notice that by the given construction $\beta_i < 1$ for all $i = 1, \ldots, n-1$ since $x_{i+1} < x_i$. In the sequel we will show that x_n is only polynomially smaller than x_1, namely $x_n = \Omega(x_1/n^4)$. We then derive a bound on the distance between any consecutive pair x_i and x_{i+1}. These two assertions combined, imply that the point set has a spread bounded by $O(n^5)$. The following lemma follows from elementary algebraic simplifications.

Lemma 2.2 *For each $1 \leq i \leq n/2$, $\beta_i \geq (1 - 1/i)^2$, and for each $n/2 < i < n-1$, $\beta_i \geq (1 - 1/(n-i+1))^2$. Furthermore, for $i \geq 2$, we have $\beta_i \leq 1 - 1/2i$.*

Corollary 2.3 *For any $n > 0$ we have $x_n = \Omega(x_1/n^4)$.*

Proof: $$x_n = \beta_1 \cdot \prod_{i=2}^{n-1} \beta_i \cdot x_1 \geq \beta_1 x_1 \cdot \prod_{i=2}^{\lfloor n/2 \rfloor} \left(1 - \frac{1}{i}\right)^2 \cdot \prod_{i=\lfloor n/2 \rfloor + 1}^{n-1} \left(1 - \frac{1}{n-i+1}\right)^2$$
$$= \beta_1 x_1 \cdot \left(1 - \frac{1}{2}\right)^2 \cdots \left(1 - \frac{1}{\lfloor n/2 \rfloor}\right)^2 \cdot \left(1 - \frac{1}{\lfloor n/2 \rfloor}\right)^2 \cdots \left(1 - \frac{1}{2}\right)^2$$
$$= \beta_1 x_1 \cdot \left(\prod_{i=2}^{\lfloor n/2 \rfloor} \frac{(i-1)^2}{i^2}\right)^2 = \beta_1 x_1 \cdot \left(\frac{1}{\lfloor n/2 \rfloor}\right)^4$$

The claim follows as $\beta_1 = n/(2n-1) = \Theta(1)$. ■

Lemma 2.4 *For each $i = 1, \ldots, n-1$, $x_i - x_{i+1} \geq x_i/3i$.*

Proof: Since $x_{i+1} = \beta_i x_i$, we have $x_i - x_{i+1} = x_i(1 - \beta_i)$. For $i = 1$, we have $\beta_1 = n/(2n-1) \leq 2/3$, when $n \geq 2$. Thus, we have $x_1 - x_2 \geq x_1/3$. For $i = 2, \ldots, n-1$, using Lemma 2.2 we get $1 - \beta_i \geq 1/2i$. Thus, $x_i - x_{i+1} = x_i(1 - \beta_i) \geq x_i \cdot 1/(2i) > x_i/3i$, as claimed. ■

Theorem 2.5 *The spread of the point set constructed in Theorem 2.1 is $O(n^5)$.*

Proof: By Lemma 2.4, for each $i = 1, \ldots, n-1$, $x_i - x_{i+1} \geq x_i/3i$. Since $x_i > x_n$ and by Corollary 2.3, $x_n = \Omega(x_1/n^4)$, it follows that $x_i - x_{i+1} = \Omega(x_1/n^5)$. This lower bound for the distance between two consecutive points is also true for y_i's due to the symmetric construction of the point set around 0. On the other hand, since $x_n = \Omega(x_1/n^4)$, $x_n - y_n = 2x_n = \Omega(x_1/n^4)$. Thus every pair of points are at distance at least $\Omega(x_1/n^5)$. Since the diameter of the point set is $2x_1$, we get a bound of $O(n^5)$ for the spread of the point set. ■

3 An Upper Bound for One Dimension

In this section, we prove an upper bound on the number of steps of k-MeansMtd in one dimensional Euclidean space. As we shall see, the bound does not involve k but is instead related to the spread Δ of the point set X. Without loss of generality we can assume that the closest pair of points in X are at distance 1 and thus the diameter of the set X is Δ. Before proving the upper bound, we mention a straightforward technical lemma from [KMN+02].

Lemma 3.1 ([KMN+02]) *Let S be a set of points in $\mathbb{R}^d$ with centroid $c = c(S)$ and let z be an arbitrary point in $\mathbb{R}^d$. Then $\psi(S,z) - \psi(S,c) = \sum_{x \in S}\left(\|x - z\|^2 - \|x - c\|^2\right) = |S| \cdot \|c - z\|^2$.*

The above lemma quantifies the contribution of a center c_i to the cost improvement in a k-Means step as a function of the distance it moves. More formally, if in a k-Means step a k-clustering $\mathcal{S} = (S_1, \ldots, S_k)$ is changed to the other k-clustering $\mathcal{S}' = (S'_1, \ldots, S'_k)$, then

$$\phi(\mathcal{S}) - \phi(\mathcal{S}') \geq \sum_{i=1}^{k} |S'_i| \cdot \left\|c(S'_i) - c(S_i)\right\|^2 .$$

Note that in the above analysis we only consider the improvement resulting from the second stage of the k-Means step in which the centers are moved to the centroids of their clusters. There is an additional gain from reassigning the points in the first stage of a k-Means step that we currently ignore.

In all our upper bound arguments we use the fact that since the initial set of centers is chosen from inside the convex hull of the input point set X (the initial centers are the centroid of the initial arbitrary clustering and even if this was not the case, all centers would move inside the convex hull of X after one step), the initial clustering cost is no more than $n\Delta^2$. This simply follows from the fact that each of the n points in X is at distance no more than Δ from its assigned center.

Theorem 3.2 *The number of steps of k-MeansMtd on a set $X \subset \mathbb{R}$ of n points with spread Δ is at most $O(n\Delta^2)$.*

Proof: Consider a k-Means step that changes a k-clustering $\mathcal{S}$ into another k-clustering $\mathcal{S}'$. The crucial observation is that in this step, there exists a cluster that is only extended or shrunk from its right end. To see this consider the leftmost cluster S_1. Either S_1 is modified in this step, in which case this modification can only happen in form of extension or shrinking at its right end, or it remains the same. In the latter case, the same argument can be made about S_2, and so on.

Thus assume that S_1 is extended from right by receiving a set T or points from the cluster directly to its right, namely S_2 (S_2 cannot lose all its points to S_1 as it has at least one point to the right of c_2 and this point is closer to c_2 than to c_1 and cannot go to S_1). Notice that $c(T)$ is to the right of the leftmost point in T and at distance at least $(|T|-1)/2$ from this leftmost point (because every pair of points are at distance one or more in T and $c(T)$ gets closest to its leftmost point when every pair of consecutive points in T are placed at the minimum distance of 1 from each other). Similarly, the centroid of S_1 is to the left of the rightmost point of S_1 and at distance at least $(|S_1|-1)/2$ from it. Thus, $\|c(S_1)-c(T)\| \geq (|T|-1)/2+(|S_1|-1)/2+1 = (|T|+|S_1|)/2$, where the extra 1 is added because the distance between the leftmost point in T and the rightmost point in S_1 is at least 1. The centroid of S'_1 will therefore be at distance

$$\frac{|T|}{|S_1|+|T|}\|c(S_1)-c(T)\| \geq \frac{|T|}{|S_1|+|T|}\cdot\frac{|T|+|S_1|}{2} = \frac{|T|}{2} \geq \frac{1}{2}$$

from $c(S_1)$ and to its right. Consequently, by Lemma 3.1, the improvement in clustering cost is at least $1/4$.

Similar analysis implies a similar improvement in the clustering cost for the case where we remove points from S_1. Since the initial clustering cost is at most $n\Delta^2$, the number of steps is no more than $n\Delta^2/(1/4) = 4n\Delta^2$. ■

Remark 3.3 The choice of $n\Delta^2$ as an upper-bound for the initial clustering cost in proving Theorem 3.2 as well as all other upper bounds proved later in this paper can be slightly improved, tightening these upper bounds accordingly.

Since the initial centers are centroids of their clusters in the initial clustering, at the beginning of the first step, we have a clustering $\mathcal{S} = (S_1,\ldots,S_k)$ of the input point set X with centers $c_1,\ldots,c_k$, respectively, where for each $i=1,\ldots,k$, $c_i = c(S_i)$. Let $\hat{c} = c(X)$ be the centroid of the entire input point set. By Lemma 3.1, we can write

$$\psi(S_i,c_i) = \psi(S_i,\hat{c}) - |S_i|\cdot\|\hat{c}-c_i\|^2\,,$$

for $1 \leq i \leq k$. Summing this equation, for $i=1,\ldots,k$, we get

$$\phi(\mathcal{S}) = \sum_{x\in X}\|x-\hat{c}\|^2 - \sum_{i=1}^{k}|S_i|\,.\|\hat{c}-c_i\|^2 < \sum_{x\in X}\|\hat{c}-x\|^2 = \frac{1}{n}\sum_{x,y\in X}\|x-y\|^2\,.$$

Thus, we get the more accurate upper bound of $1/n \sum_{x,y \in X} \|x - y\|^2$ that can replace the trivial bound of $n\Delta^2$. Note that depending on the input, this improved upper bound can be smaller than $n\Delta^2$ by a factor of $O(n)$. Nevertheless, in all our upper bound results we employ the weaker bound for the purpose of readability, while all those bounds can be made more precise by applying the above-mentioned improvement.

Remark 3.4 A slight technical detail in the implementation of k-MeansMtd algorithm, involves the event of a center losing all the points it serves. Candidate strategies used in practice to handle this problem include: placing the lonely center somewhere else arbitrarily or randomly, leaving it where it is to perhaps acquire some points in future steps, or completely removing it. For the sake of convenience in our analysis and in agreement with [Llo82], we adopt the last strategy, namely, whenever a center is left serving no points, we remove that center permanently and continue with the remaining centers.

4 Upper Bound for Points on a d-Dimensional Grid

In this section, we prove an upper bound on the number of steps of k-MeansMtd when the input points belong to the integer grid $\{1, \ldots, M\}^d$. This is the case in many practical applications where every data point has a large number of fields with each field having values in a small discrete range. For example, this includes clustering of pictures, where every pixel forms a single coordinate (or three coordinates, corresponding to the RGB values) and the value of every coordinate is restricted to be an integer in the range 0–255.

The main observation is that the centroids of any two subsets of $\{1, \ldots, M\}^d$ are either equal or are suitably far away. Since each step of k-MeansMtd moves at least one center or else stops, this observation guarantees a certain amount of improvement to the clustering cost in each step.

Lemma 4.1 *Let S_1 and S_2 be two nonempty subsets of $\{1, \ldots, M\}^d$ with $|S_1| + |S_2| \leq n$. Then, either $c(S_1) = c(S_2)$ or $\|c(S_1) - c(S_2)\| \geq 1/n^2$.*

Proof: If $c(S_1) \neq c(S_2)$ then they differ in at least one coordinate. Let u_1 and u_2 be the values of $c(S_1)$ and $c(S_2)$ in one such coordinate, respectively. By definition, $u_1 = s_1/|S_1|$ and $u_2 = s_2/|S_2|$ where s_1 and s_2 are integers in the range $\{1, \ldots, nM\}$. In other words $|u_1 - u_2|$ is the difference of two distinct fractions, both with denominators less than n. It follows that $|u_1 - u_2| \geq 1/n^2$ and consequently $\|c(S_1) - c(S_2)\| \geq |u_1 - u_2| \geq 1/n^2$. ■

Theorem 4.2 *The number of steps of k-MeansMtd when executed on a point set X taken from the grid $\{1, \ldots, M\}^d$ is at most dn^5M^2.*

Proof: Note, that $U = n \cdot (\sqrt{d}M)^2 = ndM^2$ is an upper bound of for the clustering cost of any k-clustering of a point set in $\{1, \ldots, M\}^d$ and that at each step at least one center moves by at least $1/n^2$. Therefore, by Lemma 3.1, at every step the cost function decreases by at least $1/n^4$ and the overall number of steps can be no more than $U/(1/n^4) = dn^5M^2$. ■

5 Arbitrary Point Sets and Alternative Algorithms

Unfortunately proving any meaningful bounds for the general case of k-MeansMtd, namely with points in $\mathbb{R}^d$ with $d > 1$ and no further restrictions, remains elusive. However, in this section, we present two close relatives of k-MeansMtd for each of which we can bound the number of steps by a polynomial in the number of points, number of centers, and the spread of the point set. The first algorithm differs from k-MeansMtd in that it moves a misclassified point to its correct cluster, as soon as the misclassified point is discovered (rather than first finding all misclassified points and then reassigning them to their closest centers as is the case in k-MeansMtd). The second algorithm is basically the same as k-MeansMtd using a natural generalization of misclassified points. Our experimental results (Section 6) further support the kinship of these two algorithms with k-MeansMtd.

As was the case with our previous upper bounds, our main approach in bounding the number of steps in both these algorithms is through showing substantial improvements in the clustering cost at each step.

5.1 The SinglePnt Algorithm

We introduce an alternative to the k-Means step which we shall call a SinglePnt step.

Definition 5.1 In a SinglePnt *step* on a k-clustering $\mathcal{S} = (S_1, \ldots, S_k)$, a misclassified point x is chosen, such that $x \in S_i$ and $\|x - c(S_j)\| < \|x - c(S_i)\|$, for some $1 \le i \ne j \le k$, and a new clustering $\mathcal{S}' = (S'_1, \ldots, S'_k)$ is formed by removing x from S_i and adding it to S_j. Formally, for each $1 \le l \le k$,

$$S'_l = \begin{cases} S_l & l \ne i, j, \\ S_l \setminus \{x\} & l = i, \\ S_l \cup \{x\} & l = j. \end{cases}$$

The centers are updated to the centroids of the clusters, and therefore only the centers of S_i and S_j change. Note that updating the centers takes constant time.

In a SinglePnt step, if the misclassified point is far away from at least one of $c(S_i)$ and $c(S_j)$, then the improvement in clustering cost made in the SinglePnt step cannot be too small.

Lemma 5.2 *Let S and T be two point sets of sizes n and m, respectively, and let $s = c(S)$ and $t = c(T)$. Suppose that x is a point in T with distances d_S and d_T from s and t, respectively, and such that $d_S < d_T$. Let $S' = S \cup \{x\}$ and $T' = T \setminus \{x\}$ and let $s' = c(S')$ and $t' = c(T')$. Then $\psi(S, s) + \psi(T, t) - \psi(S', s') - \psi(T', t') \ge (d_S + d_T)^2/(2(n + m))$.*

Proof: Indeed, $c(S') = \frac{n}{n+1}c(S) + \frac{1}{n+1}x$. Thus

$$\|s - s'\| = \|c(S) - c(S')\| = \left\| \frac{1}{n+1}c(S) - \frac{1}{n+1}x \right\| = \frac{1}{n+1}\|c(S) - x\| = \frac{d_S}{n+1}.$$

Similarly, $\|t - t'\| = d_T/(m - 1)$. By Lemma 3.1 we obtain

$$\psi(S', s) - \psi(S', s') = (n + 1)\left(\frac{d_S}{n+1}\right)^2 = \frac{d_S^2}{n+1},$$

and similarly $\psi(T', t) - \psi(T', t') = d_T^2/(m - 1)$.

Since $d_S < d_T$, we have that $\psi(S, s) + \psi(T, t) \ge \psi(S', s) + \psi(T', t)$, and

$$\begin{aligned} \psi(S, s) + \psi(T, t) - \psi(S', s') - \psi(T', t') &\ge \psi(S', s) + \psi(T', t) - \psi(S', s') - \psi(T', t') \\ &\ge \frac{d_S^2}{n+1} + \frac{d_T^2}{m-1} \ge \frac{d_S^2}{n+m} + \frac{d_T^2}{n+m} = \frac{d_S^2 + d_T^2}{n+m} \ge \frac{(d_S + d_T)^2}{2(n+m)}. \end{aligned}$$

■

Our modified version of k-MeansMtd, to which we shall refer as "SinglePnt", replaces k-Means steps with SinglePnt steps. Starting from an arbitrary clustering of the input point set, SinglePnt repeatedly performs SinglePnt steps until no misclassified points remain. Notice that unlike the k-Means step, the SinglePnt step does not maintain the property that the clustering achieved at the end of the step is imposed by some Voronoi diagram. However, this property must hold when the algorithm stops, or otherwise some misclassified points would exist and further steps would be possible.

Theorem 5.3 *On any input $X \subset \mathbb{R}^d$, SinglePnt makes at most $O(kn^2\Delta^2)$ steps before termination.*

Proof: Once again, we assume that no two points in X are less than unit distance apart. Call a SinglePnt step *weak*, if the misclassified point it considers is at distance less than $1/8$ from both *involved centers*, i.e., its current center and the center closest to it. We call a SinglePnt step *strong* if it is not weak. Since in a strong SinglePnt step, the sum of distances of the misclassified point to the involved centers is at least $1/8$, and the two involved clusters have at most n points combined, it follows by Lemma 5.2 that in such a step the clustering cost improves by at least $(1/8)^2/(2n) = 1/(128n)$. In the sequel we shall show that the algorithm cannot take more than k *consecutive* weak steps, and thus at least one out of every $k+1$ consecutive steps must be strong and thus result an improvement of $1/(128n)$ to the clustering cost; hence the upper bound of $O(kn^2\Delta^2)$.

Consider the beginning of a of SinglePnt step. Let $c_1, \ldots, c_k$ denote the current centers, and let $S_1, \ldots, S_k$ denote the corresponding clusters; namely, S_i is the set of points served by c_i, for $i = 1, \ldots, k$. Consider the balls $B_1, \ldots, B_k$ of radius $1/8$ centered at $c_1, \ldots, c_k$, respectively. Observe that since every pair of points in X are at distance at least 1 from each other, each ball B_i can contain at most one point of X. Moreover, the intersection of any subset of the balls $B_1, \ldots, B_k$ can contain at most one point of X. For a point $x \in X$, let $\mathcal{B}(x)$ denote the set of balls among $B_1, \ldots, B_k$ that contain the point x. We refer to $\mathcal{B}(x)$ as the *batch* of x.

By the above observation, the balls (and the corresponding centers) are classified according to the point of X they contain (if they contain such a point at all). Let $\mathcal{B}_X$ be the set of batches of balls that are induced by X and contain more than one ball. Formally, $\mathcal{B}_X = \{\mathcal{B}(x) : x \in X, |\mathcal{B}(x)| > 1\}$. The set of balls $\bigcup \mathcal{B}_X$ is the set of *active* balls.

A misclassified point x can participate in a weak SinglePnt step only if it belongs to more than one ball; i.e., when $|\mathcal{B}(x)| > 1$. Observe that, if we perform a weak step, and one of the centers moves such that the corresponding ball B_i no longer contains any point of X in its interior, then for B_i to contain a point again, the algorithm must perform a strong step. To see this, observe that (weakly) losing a point x may cause a center move a distance of at most $1/8$. Therefore, once a center c_i loses a point x, and thus moves away from x, it does not move far enough for the ball B_i to contain a different point of X.

Hence, in every weak iteration a point x changes the cluster it belongs to in $\mathcal{B}(x)$. This might result in a shrinking of the active set of balls. On the other hand, while only weak SinglePnt steps are being taken, any cluster S_j can change only by winning or losing the point x_i that stabs the corresponding ball B_j. It follows that once a set S_j loses the point x, then it can never get it back since that would correspond to an increase in the clustering cost. Therefore the total number of possible consecutive weak SinglePnt steps is bounded by $\sum_{x \in X, |\mathcal{B}(x)|>1} |\mathcal{B}(x)| \leq k$. ■

5.2 The Lazy-k-Means algorithm

Our second variant to k-MeansMtd, which we name "Lazy-k-Means", results from a natural generalization of misclassified points (Definition 1.1). Intuitively, the difference between the Lazy-

k-Means and k-MeansMtd is that Lazy-k-Means at each step only reassigns those misclassified points to their closest centers that are *substantially* misclassified, namely the points that benefit from reclassification by at least a constant factor.

Definition 5.4 Given a clustering $\mathcal{S} = (S_1, \dots, S_k)$ of a point set X, if for a point $x \in S_i$ there exists a $j \neq i$, such that $\|x - c(S_i)\| > (1+\varepsilon)\|x - c(S_j)\|$, then x is said to be $(1+\varepsilon)$-*misclassified* for center pair $(c(S_i), c(S_j))$. The centers $c(S_i)$ and $c(S_j)$ are referred to as *switch centers* for x. We also say that $c(S_i)$ is the *losing center* and $c(S_j)$ is the *winning center* for x.

Thus Lazy-k-Means with parameter ε starts with an arbitrary k-clustering. In each step, it (i) reassigns every $(1+\varepsilon)$-misclassified point to its closest center and (ii) moves every center to the centroid of its new cluster. Indeed, k-MeansMtd is simply Lazy-k-Means with parameter $\varepsilon = 0$. Naturally, the algorithm stops when no $(1+\varepsilon)$-misclassified points are left.

In the sequel we bound the maximum number of steps taken by Lazy-k-Means. We shall use the following fact from elementary Euclidean geometry.

Fact 5.5 *Given two points c and c' with $\|c - c'\| = \ell$, the locus of the points x with $\|x - c'\| > (1+\varepsilon)\|x - c\|$ is an open ball of radius $R = \ell(1+\varepsilon)/(\varepsilon(2+\varepsilon))$ called the ε-Apollonius ball for c with respect to c'. This ball is centered on the line containing the segment cc' at distance $R + \ell\varepsilon/(2(2+\varepsilon))$ from the bisector of cc', and on the same side of the bisector as c.*

Lemma 5.6 *For any three points x, c, and c' in $\mathbb{R}^d$ with $\|x - c\| \le \|x - c'\|$ we have $\|x - c'\|^2 - \|x - c\|^2 = 2h\|c - c'\|$, where h is the distance from x to the bisector of c and c'.*

Proof: Let y be the intersection point of the segment cc' with the $(d-1)$-dimensional hyperplane parallel to the bisector of c and c' and containing x. By Pythagorean equality we have $\|x - c\|^2 = \|x - y\|^2 + \|y - c\|^2$ and $\|x - c'\|^2 = \|x - y\|^2 + \|y - c'\|^2$. Subtracting the first equality from the second, we obtain

$$\begin{aligned} \|x - c'\|^2 - \|x - c\|^2 &= \|y - c'\|^2 - \|y - c\|^2 \\ &= (\|y - c'\| + \|y - c\|)(\|y - c'\| - \|y - c\|) \\ &= 2h\|c - c'\|, \end{aligned}$$

since $\|y - c'\| - \|y - c\| = 2h$. ■

Theorem 5.7 *For $\varepsilon > 0$, the number of steps of* Lazy-k-Means *is $O(n\Delta^2\varepsilon^{-3})$.*

Proof: We will show that every two consecutive steps of Lazy-k-Means make an improvement of at least

$$\lambda^* = \frac{\varepsilon^3(2+\varepsilon)}{256(1+\varepsilon)^2} \ge \frac{\varepsilon^3}{512} = \Omega(\varepsilon^3).$$

Let $\ell_0 = \varepsilon(2+\varepsilon)/(16(1+\varepsilon))$. Notice that $\ell_0 < 1/8$ for $0 < \varepsilon \le 1$. We call a misclassified point x *strongly* misclassified, if its switch centers c and c' are at distance at least ℓ_0 from each other, and *weakly* misclassified otherwise.

If at the beginning of a Lazy-k-Means step there exists a strongly misclassified point x for a center pair (c, c'), then since every point in the ε-Apollonius ball for c' with respect to c is at distance at least $\ell_0\varepsilon/(2(2+\varepsilon))$ from the bisector of cc', by Lemma 5.6 the reclassification improvement in clustering cost resulting from assigning x to c' is

$$\|x - c\|^2 - \|x - c'\|^2 = \frac{\ell_0^2\varepsilon}{2+\varepsilon} \ge \frac{\varepsilon^3(2+\varepsilon)}{256(1+\varepsilon)^2} = \lambda^*.$$

Thus we assume that all misclassified points are weakly misclassified. Let x be one such point for center pair (c, c'). By our assumption $\|c - c'\| < \ell_0$. Observe that in such a case, the radius of the ε-Apollonius ball for c' with respect to c is $\ell(1+\varepsilon)/(\varepsilon(2+\varepsilon)) < 1/16$. In particular, since there exists a ball of radius 1/16 containing both x and c', the ball of radius 1/8 centered at c', which we denote by $B(c', 1/8)$, includes x. Also since $\|c - c'\| < 1/8$ as verified above, we get $c \in B(c', 1/8)$ as well. In other words, both switch centers c and c' are at distance less than 1/4 from x. Now, since every pair of points in X are at distance 1 or more, any center can be a switch center for at most one weakly misclassified point. This in particular implies that in the considered Lazy-k-Means step, no cluster is modified by more than a single point.

When the misclassified points are assigned to their closest centers, the centers that do not lose or win any points stay at their previous locations. A center c' that wins a point x moves closer to x since x is the only point it wins while losing no other points. Similarly, a center c that loses a point x moves away from x since x is the only point it loses without winning any other points. A losing center c moves away from its lost point x by a distance of at most $\|c - x\| < 1/4$ since its previous number of served points was at least 2 (otherwise, we would have $c = x$ and thus x could not be misclassified). Therefore, when c moves to the centroid of its cluster (now missing x), $\|x - c\| < 1/2$ and consequently $\|c - y\| > 1/2$ for any $x \neq y \in X$. As a result, c can not be a switch center for any weakly misclassified point in the subsequent Lazy-k-Means step.

On the other hand, the winning center c' to whose cluster x is added, moves closer to x and since no center other than c and c' in $B(x, 1/4)$ moves (as there is no point other than x they can win or lose), x will not be misclassified in the next Lazy-k-Means step.

It follows from the above discussion that the next Lazy-k-Means step cannot have any weakly misclassified points and thus either the algorithm stops or some strongly misclassified point will exist, resulting an improvement of at least λ^*. Thus the total number of steps taken by Lazy-k-Means with parameter ε is at most $2n\Delta^2/\lambda^* = O(n\Delta^2\varepsilon^{-3})$. ■

6 Experimental Results

We introduced both SinglePnt and Lazy-k-Means alternatives to k-MeansMtd as similar, equally easy to implement algorithms that are simpler to analyze than k-MeansMtd itself. However, as mentioned in the introduction, k-MeansMtd is mainly of interest only in practice because of its ease of implementation and its relatively fast termination (small number of steps). It thus raises the question of how our alternative algorithms perform in practice in comparison to k-MeansMtd.

We performed a series of experiments analogous to those done in [KMN+02], as described below, to compare the number of rounds, number of reclassified points, and quality of final clustering produced by these two alternative algorithms with those of k-MeansMtd. We use the same inputs used by Kanungo *et al.* for our experiments. See [KMN+02] for detailed description of those inputs. We have tried to implement each of the algorithms in the simplest possible way and avoided using any advanced point location or nearest neighbor search structure or algorithm. Due to the great similarity between the three algorithms considered here, it is expected that any technique used for improving the performance of any of these algorithms, to be suitable for improving the other two variants in a somewhat similar way.

The algorithms k-MeansMtd and Lazy-k-Means iterate over points and assign each point to the closest center. While doing this the new set of centers are calculated and existence of a $(1 + \varepsilon)$-misclassified point is checked. SinglePnt examines the points one by one, moving to the first point when reaching the end of the list, checking if they are misclassified or not. When a misclassified point is discovered it is assigned to its closest center and the location of the two switching centers is updated. The algorithm stops when it cannot find a misclassified point for n consecutive steps.

The input used in these experiments together with the source-code of our implementation is available at [Sad04].

Our experimental results are summarized in Table 1 and Table 2. In conformance to [KMN+02] the costs referred to in these tables is the total final clustering cost, divided by the number of points. In that sense we report the "average" cost per point. Table 1 is produced by running, only once, each of the four algorithms with the same set of randomly chosen center for each combination of point set and number of centers considered. By studying several such tables it seems that the total number of reclassified points and the quality of clustering found by SinglePnt tends to be very close to those of k-MeansMtd. Notice that in Table 1, the number of steps of SinglePnt are left blank as they are equal to the number of reclassified points and cannot be compared with the number of steps of k-MeansMtd or Lazy-k-Means.

Table 2 summarizes the results of running 100 tests similar to the one reported in Table 1 each with different initial set of centers picked randomly from the bounding box of the given point set. The best, worst, and average final clustering costs are reported in each case.

We have not discussed the running times as we made no effort in optimizing our implementations. It is however interesting that both of the two alternative algorithms tend to be faster than k-MeansMtd's in a typical implementation such as ours. SinglePnt seems to be typically more than 20% faster than k-MeansMtd. In particular, we emphasize, that our simple implementation is considerably slower than the implementation of Kanungo *et al.* [KMN+02] that uses data structure similar to *kd*-tree to speed up the computation of the Voronoi partitions. We believe that we would get similar performance gains by using their data structure.

7 Conclusions

We presented several results on the number of iterations performed by the k-MeansMtd clustering algorithm. To our knowledge, our results are the first to provide combinatorial bounds on the performance of k-MeansMtd. We also suggested related variants of k-MeansMtd algorithm, and proved upper bounds for their performance. We implemented those algorithms and compared their performance in practice [Sad04]. We conjecture that the upper bounds we proved for SinglePnt hold also for k-MeansMtd. Maybe the most surprising part in those bounds for the number of iterations performed is the lack of dependence on the dimension of the data.

We consider this paper to be a first step in understanding the k-means method. It is our belief that both our lower and upper bounds are loose, and one might need to use other techniques to improve them. In particular, we mention some open problems:

1. There is still a large gap between our lower and upper bounds. In particular, a super-linear lower bound would be interesting even in high-dimensional space.

2. Our current upper bounds include the spread as a parameter. It would be interesting to prove (or disprove) that this is indeed necessary.

3. We have introduced alternative, easy to analyze algorithms, that are comparable to k-MeansMtd both in their description and their behavior in practice. It would be interesting to show provable connections between these algorithms and compare the bounds on the number of steps they require to terminate.

7.1 Dependency on the spread

A shortcoming of our results, is the dependency on the spread of the point set in the bounds presented. However:

1. This can be resolved by doing a preprocessing stage, snapping together points close to each other, and breaking the input into several parts to be further clustered separately. This is essentially what fast provable approximation algorithms for TSP, k-means, and k-median do [Aro98, HM04]. This results in point sets with polynomial spread, which can be used instead of the original input to compute a good clustering. This is outside the scope of our analysis, but it can be used in practice to speedup k-MeansMtd algorithm.

2. In high dimensions, it seems that in many natural cases the spread tends to shrink and be quite small. As such, we expect our bounds to be meaningful in such cases.

   To see an indication of this shrinkage in the spread, imagine picking n points randomly from a unit hypercube in $\mathbb{R}^d$ with volume one. It is easy to see that the minimum distance between any pair of points is going to be at least $L = 1/n^{3/d}$, with high probability, since if we center around each such point a hypercube of side length L, it would have volume $1/n^3$ of the unit hypercube. As such, the probability of a second point falling inside this region is polynomially small.

   However, L tends to 1 as d increases. Thus, for $d = \Theta(\log n)$ the spread of such random point set is $\Theta(\sqrt{d}/(L/2)) = \Theta(\sqrt{\log n})$. (An alternative way to demonstrate this is by picking points randomly from the unit hypersphere. By using a concentration of mass argument [Mat02] on a hypersphere, we get a point-set with spread $O(1)$ with high probability.)

7.2 Dependency on the initial solution

The initial starting solution fed into k-MeansMtd is critical in the time it takes to converge, and in the quality of the final clustering generated. This is clearly suggested by Table 2, where trying many different initial solutions has yielded a considerable improvement in the best found solution. Of course, one can use a (rough) approximation algorithm [HM04] to come up with a better starting solution. While this approach might be useful in practice, it again falls outside the scope of our analysis.

In particular, it would be nice to improve our lower bound, so that it holds, with reasonable probability, for randomly chosen initial solution.

7.3 Similar results

Recently, independently of our results, Sanjoy Dasgupta [Das03] announced results which are similar to a *subset* of our results. In particular, he mentions the one-dimensional lower bound, and a better upper bound for $k < 5$ but only in one dimension. This work of Sanjoy Dasgupta and Howard Karloff seems to be using similar arguments to ours (personal communication) although to our knowledge it has not been written or published yet.

Acknowledgments

The authors would like to thank Pankaj K. Agarwal, Boris Aronov and David Mount for useful discussions of problems studied in this paper and related problems. In particular, David Mount provided us with the test point sets used in [KMN+02]. The authors would also like to thank the referees for their comments.

References

[Aro98] S. Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5):753–782, Sep 1998.

[Das03] S. Dasgupta. How fast is k-means? In *Proc. 16th Annu. Comp. Learn. Theo.*, number 2777 in Lect. Notes in Comp. Sci., page 735, 2003.

[DFG99] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2nd edition, 2001.

[dlVKKR03] W. F. de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. In *Proc. 35th Annu. ACM Sympos. Theory Comput.*, pages 50–58, 2003.

[ES03] M. Effros and L. J. Schulman. Deterministic clustering with data nets. Manuscript, 2003.

[HM04] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. In *Proc. 36th Annu. ACM Sympos. Theory Comput.*, pages 291–300, 2004.

[IKI94] M. Inaba, N. Katoh, and H. Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 332–339, 1994.

[KMN+02] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 10–18, 2002.

[KSS04] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proc. 45th Annu. IEEE Sympos. Found. Comput. Sci.*, page to appear, 2004.

[Llo82] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[Mac67] J. MacQueen. Some methods for classifications and analysis of multivariate observations. In *Proc. fifth Berkeley symp. math. stat. and prob.*, pages 281–297. Unversity of California Press, Berkeley, 1967.

[Mat00] J. Matoušek. On approximate geometric k-clustering. *Discrete Comput. Geom.*, 24:61–84, 2000.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[Pol81] D. Pollard. Strong consistency of k-means clustering. *Annals of Statistics*, 9:135–140, 1981.

[Sad04] B. Sadri. Lloyd's method and variants implementation together with inputs, 2004. http://www.uiuc.edu/~sariel/papers/03/lloyd_kmeans.

Data Set	k	Method	Steps	Reclassified	Final Cost
ClusGauss $n = 10,000$ $d = 3$	25	k-MeansMtd	24	4748	0.081615
		SinglePnt	-	4232	0.081622
		Lazy-k-Means, $\epsilon = 0.05$	17	2377	0.082702
		Lazy-k-Means, $\epsilon = 0.20$	18	1554	0.089905
	50	k-MeansMtd	20	4672	0.031969
		SinglePnt	-	4391	0.031728
		Lazy-k-Means, $\epsilon = 0.05$	16	2244	0.032164
		Lazy-k-Means, $\epsilon = 0.20$	22	1974	0.034661
	100	k-MeansMtd	22	5377	0.009639
		SinglePnt	-	4958	0.009706
		Lazy-k-Means, $\epsilon = 0.05$	15	2512	0.010925
		Lazy-k-Means, $\epsilon = 0.20$	19	1748	0.013092
MultiClus $n = 10,000$ $d = 3$	50	k-MeansMtd	21	2544	0.033870
		SinglePnt	-	2419	0.033941
		Lazy-k-Means, $\epsilon = 0.05$	16	1121	0.034622
		Lazy-k-Means, $\epsilon = 0.20$	25	722	0.038042
	100	k-MeansMtd	18	1744	0.009248
		SinglePnt	-	1732	0.008854
		Lazy-k-Means, $\epsilon = 0.05$	11	740	0.009902
		Lazy-k-Means, $\epsilon = 0.20$	15	584	0.010811
	500	k-MeansMtd	12	1768	0.002495
		SinglePnt	-	1694	0.002522
		Lazy-k-Means, $\epsilon = 0.05$	9	528	0.002757
		Lazy-k-Means, $\epsilon = 0.20$	11	444	0.002994
Lena22 $n = 65,536$ $d = 4$	8	k-MeansMtd	36	62130	335.408625
		SinglePnt	-	57357	335.440866
		Lazy-k-Means, $\epsilon = 0.05$	27	50298	338.594668
		Lazy-k-Means, $\epsilon = 0.20$	21	44040	355.715258
	64	k-MeansMtd	211	111844	94.098422
		SinglePnt	-	81505	94.390640
		Lazy-k-Means, $\epsilon = 0.05$	88	55541	97.608823
		Lazy-k-Means, $\epsilon = 0.20$	24	30201	120.274428
	256	k-MeansMtd	167	111110	48.788216
		SinglePnt	-	101522	48.307815
		Lazy-k-Means, $\epsilon = 0.05$	92	57575	51.954810
		Lazy-k-Means, $\epsilon = 0.20$	79	32348	61.331614
Lena44 $n = 16,384$ $d = 16$	8	k-MeansMtd	63	18211	2700.589245
		SinglePnt	-	16467	2700.587691
		Lazy-k-Means, $\epsilon = 0.05$	20	9715	2889.747540
		Lazy-k-Means, $\epsilon = 0.20$	27	9201	3008.783333
	64	k-MeansMtd	61	21292	1525.846646
		SinglePnt	-	16422	1615.667299
		Lazy-k-Means, $\epsilon = 0.05$	45	13092	1555.520952
		Lazy-k-Means, $\epsilon = 0.20$	16	7527	1907.962692
	256	k-MeansMtd	43	21394	1132.746162
		SinglePnt	-	28049	1122.407317
		Lazy-k-Means, $\epsilon = 0.05$	28	12405	1156.884049
		Lazy-k-Means, $\epsilon = 0.20$	27	7993	1320.303278
Kiss $n = 10,000$ $d = 3$	8	k-MeansMtd	18	5982	687.362264
		SinglePnt	-	7026	687.293930
		Lazy-k-Means, $\epsilon = 0.05$	18	3277	690.342895
		Lazy-k-Means, $\epsilon = 0.20$	23	2712	720.891998
	64	k-MeansMtd	202	29288	202.044849
		SinglePnt	-	35228	185.519927
		Lazy-k-Means, $\epsilon = 0.05$	92	12471	221.936175
		Lazy-k-Means, $\epsilon = 0.20$	44	6080	263.497185
	256	k-MeansMtd	144	17896	105.438490
		SinglePnt	-	16992	106.112133
		Lazy-k-Means, $\epsilon = 0.05$	61	7498	120.317362
		Lazy-k-Means, $\epsilon = 0.20$	27	3479	150.156231

Table 1: Number of steps, number of reclassified points, and final average clustering cost in a typical execution of each of the four algorithms on data sets mentioned in [KMN+02].

Data Set	k	Method	Minimum Cost	Maximum Cost	Average Cost
ClusGauss $n = 10,000$ $d = 3$	25	k-MeansMtd	0.068462	0.087951	0.07501276
		SinglePnt	0.067450	0.083194	0.07486010
		Lazy-k-Means, $\varepsilon = 0.20$	0.074667	0.100035	0.08510598
		Lazy-k-Means, $\varepsilon = 0.05$	0.070011	0.092658	0.07803375
	50	k-MeansMtd	0.028841	0.040087	0.03335312
		SinglePnt	0.028376	0.040623	0.03308624
		Lazy-k-Means, $\varepsilon = 0.20$	0.031175	0.046528	0.03719264
		Lazy-k-Means, $\varepsilon = 0.05$	0.029626	0.040811	0.03384180
	100	k-MeansMtd	0.011425	0.016722	0.01401549
		SinglePnt	0.010106	0.017986	0.01365492
		Lazy-k-Means, $\varepsilon = 0.20$	0.011928	0.022015	0.01565268
		Lazy-k-Means, $\varepsilon = 0.05$	0.011730	0.020600	0.01442575
MultiClus $n = 10,000$ $d = 3$	50	k-MeansMtd	0.027563	0.034995	0.03051698
		SinglePnt	0.027412	0.034167	0.03083110
		Lazy-k-Means, $\varepsilon = 0.20$	0.029507	0.055160	0.03620397
		Lazy-k-Means, $\varepsilon = 0.05$	0.028457	0.046314	0.03260643
	100	k-MeansMtd	0.002477	0.004324	0.00308144
		SinglePnt	0.002390	0.004179	0.00303798
		Lazy-k-Means, $\varepsilon = 0.20$	0.002758	0.005175	0.00356282
		Lazy-k-Means, $\varepsilon = 0.05$	0.002331	0.004789	0.00322593
	500	k-MeansMtd	0.002142	0.002731	0.00240768
		SinglePnt	0.002136	0.002805	0.00244548
		Lazy-k-Means, $\varepsilon = 0.20$	0.002539	0.003567	0.00292354
		Lazy-k-Means, $\varepsilon = 0.05$	0.002206	0.002890	0.00254321
Lena22 $n = 65,536$ $d = 4$	8	k-MeansMtd	263.644420	348.604787	299.78905632
		SinglePnt	263.659829	348.527023	307.12394164
		Lazy-k-Means, $\varepsilon = 0.20$	278.337133	414.679356	345.07986265
		Lazy-k-Means, $\varepsilon = 0.05$	271.041374	409.802396	322.99259307
	64	k-MeansMtd	82.074376	102.327255	88.53558757
		SinglePnt	82.190945	104.574941	89.24323986
		Lazy-k-Means, $\varepsilon = 0.20$	100.601485	147.170657	111.93562151
		Lazy-k-Means, $\varepsilon = 0.05$	82.798308	106.231864	94.20319250
	256	k-MeansMtd	44.637740	51.482531	47.66542537
		SinglePnt	44.699224	51.685618	47.81799127
		Lazy-k-Means, $\varepsilon = 0.20$	56.906620	71.491475	62.00216985
		Lazy-k-Means, $\varepsilon = 0.05$	47.178425	54.946136	50.82872342
Lena44 $n = 16,384$ $d = 16$	8	k-MeansMtd	2699.721266	3617.282065	2903.30164756
		SinglePnt	2699.663310	3216.854024	2894.42713876
		Lazy-k-Means, $\varepsilon = 0.20$	2834.438965	4452.875383	3293.73084140
		Lazy-k-Means, $\varepsilon = 0.05$	2725.907276	3649.518829	2977.33094524
	64	k-MeansMtd	1305.357406	1694.965827	1503.17431782
		SinglePnt	1345.821487	1811.663769	1515.08195678
		Lazy-k-Means, $\varepsilon = 0.20$	1564.252624	2385.794013	1785.93841955
		Lazy-k-Means, $\varepsilon = 0.05$	1410.883673	1793.704755	1565.18092988
	256	k-MeansMtd	1044.017122	1311.942456	1151.64441691
		SinglePnt	1055.788028	1308.459754	1168.30843808
		Lazy-k-Means, $\varepsilon = 0.20$	1262.487865	1653.820840	1400.49905496
		Lazy-k-Means, $\varepsilon = 0.05$	1094.884884	1385.345314	1219.27000492
Kiss $n = 10,000$ $d = 3$	8	k-MeansMtd	687.278119	714.789442	700.352315760
		SinglePnt	687.279479	714.731416	697.292832560
		Lazy-k-Means, $\varepsilon = 0.20$	727.017538	947.779405	802.256735040
		Lazy-k-Means, $\varepsilon = 0.05$	689.779010	861.853344	719.140385820
	64	k-MeansMtd	158.607749	208.946701	178.21703676
		SinglePnt	151.642447	203.102940	177.17793706
		Lazy-k-Means, $\varepsilon = 0.20$	222.646398	324.435479	259.62118455
		Lazy-k-Means, $\varepsilon = 0.05$	170.571861	248.648363	208.64482062
	256	k-MeansMtd	96.272602	115.294309	105.30212380
		SinglePnt	97.141907	125.009357	107.08187899
		Lazy-k-Means, $\varepsilon = 0.20$	124.378185	158.922757	140.72908431
		Lazy-k-Means, $\varepsilon = 0.05$	103.672482	129.685819	116.73971102

Table 2: Minimum, maximum, and average clustering cost on 100 executions of each of the algorithms on each of the data sets with initial centers picked randomly.