Linear-Time Approximation Scheme for k-Means Clustering of Axis-Parallel Affine Subspaces

Kyungjin Cho
POSTECH, Pohang, South Korea

Eunjin Oh
POSTECH, Poahng, South Korea

Abstract

In this paper, we present a linear-time approximation scheme for k-means clustering of *incomplete* data points in d-dimensional Euclidean space. An *incomplete* data point with $\Delta > 0$ unspecified entries is represented as an axis-parallel affine subspace of dimension Δ. The distance between two incomplete data points is defined as the Euclidean distance between two closest points in the axis-parallel affine subspaces corresponding to the data points. We present an algorithm for k-means clustering of axis-parallel affine subspaces of dimension Δ that yields an $(1+\epsilon)$-approximate solution in $O(nd)$ time. The constants hidden behind $O(\cdot)$ depend only on Δ, ϵ and k. This improves the $O(n^2 d)$-time algorithm by Eiben et al. [SODA'21] by a factor of n.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases k-means clustering, affine subspaces

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2021.46

Related Version *Full Version*: https://arxiv.org/abs/2106.14176

Funding This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2020R1C1C1012742).

1 Introduction

Clustering is a fundamental research topic in computer science, which arises in various applications [13], including pattern recognition and classification, data mining, image analysis, and machine learning. In clustering, the objective is to group a set of data points into clusters so that the points from the same cluster are similar to each other. Usually, input points lie in a high-dimensional space, and the similarity between two points is defined as their distance. Two of the popular clusterings are k-median and k-means clusterings. In the k-means clustering problem, we wish to partition a given point set into k clusters to minimize the sum of squared distances of each point to its cluster center. Similarly, in the k-median clustering problem, we wish to partition a given point set into k clusters to minimize the sum of distances of each point to its cluster center.

In this paper, we consider clustering for *incomplete data points*. The analysis of incomplete data is a long-standing challenge in practical statistics. There are lots of scenarios where entries of points of a given data set are incomplete [2]. For instance, a few questions are left blank on a questionnaire; weather records for a region omit the figures for one weather station for a short period because of a malfunction; stock exchange data is absent for one stock on one day because of a trading suspension. Various heuristic, greedy, convex optimization, statistical, or even ad hoc methods were proposed throughout the years in different practical domains to handle missing data [2].

Gao et al. [10] introduced a geometric approach to deal with incomplete data points for clustering problems. An *incomplete* point has one or more unspecified entries, which can be represented as an axis-parallel affine subspace. The distance between two incomplete data

points is defined as the Euclidean distance between two closest points in the axis-parallel affine subspaces corresponding to the data points. Since the distance between an axis-parallel affine subspace and a point is well-defined, the classical clustering problems such as k-means, k-median, and k-center can be defined on a set of axis-parallel affine subspaces.

The k-center problem in this setting was studied by [10, 11, 15]. Gao et al. [10, 11] focused on the k-center clustering for $k \leq 3$, and presented an approximation algorithm for the k-center clustering of axis-parallel affine subspaces. Later, Lee and Schulman [15] improved the running time of the algorithm by Gao et al., and then presented an $O(nd)$-time approximation algorithm for the k-center clustering problem for a larger k. The constant hidden behind $O(\cdot)$ depends on Δ, ϵ and k. Moreover, they showed that the running time of an approximation algorithm with any approximation ratio cannot be polynomial in even one of k and Δ unless P = NP, and thus the running time of their algorithm is almost tight.

Very recently, Eiben et al. [7] presented an approximation algorithm for the k-means clustering of n axis-parallel affine subspaces of dimension Δ. Their algorithm yields an $(1+\epsilon)$-approximate solution in $O(n^2 d)$ time with probability $O(1)$. The constant hidden behind $O(\cdot)$ depends on Δ, ϵ and k. Since the best-known algorithm for the k-center clustering in this setting runs in time linear in both n and d (but exponential in both k and Δ), it is a natural question if a similar time bound can be achieved for the k-means clustering. In this paper, we resolve this natural question by presenting an $(1+\epsilon)$-approximation algorithm for the k-means clustering problem running in time linear in n and d.

Related work. The k-median and k-means clustering problems for *points* in d-dimensional Euclidean space have been studied extensively. Since these problems are NP-hard even for $k = 2$ or $d = 2$ [3, 16, 18], the study of k-means and k-median clusterings have been devoted to obtain $(1+\epsilon)$-approximation algorithms for these problems [1, 5, 8, 12, 14]. These algorithms run in time polynomial time in the input size if one of k and d is constant. Indeed, it is NP-hard to approximate Euclidean k-means clustering within a factor better than a certain constant larger than one [4]. That is, the k-means clustering problem does not admit a PTAS for arbitrary k and d unless P=NP.

Also, the clustering problems for *lines* (which are not necessarily axis-parallel) also have been studied [17, 19]. More specifically, Ommer and Malik [19] presented a heuristic for k-median clustering of lines in three-dimensional space. Later, Marom and Feldman [17] presented an algorithm for computing a coreset of size $dk^{O(k)} \log n/\epsilon^2$, which gives a polynomial-time $(1+\epsilon)$-approximation algorithm for the k-means clustering of lines in d-dimensional Euclidean space.

Our results. We present an algorithm for k-means clustering of axis-parallel affine subspaces of dimension Δ that yields an $(1+\epsilon)$-approximate solution in $2^{O(\frac{\Delta^4 k}{\epsilon}(\log \frac{\Delta}{\epsilon}+k))}dn$ time with a constant probability. This improves the previously best-known algorithm by Eiben et al [7], which takes $2^{O(\frac{\Delta^7 k^3}{\epsilon}(\log \frac{k\Delta}{\epsilon}))}dn^2$ time. Since it is a generalization of the k-means clustering problem for points ($\Delta = 0$), it does not admit a PTAS for arbitrary k and d unless P=NP. Similarly to Lee and Schulman [15], we show in the full version of this paper that an approximation algorithm with any approximation ratio cannot run in polynomial time in even one of k and Δ unless P=NP. The running time of our algorithm is almost tight in the sense that it is linear in nd and exponential in k and Δ.

▶ **Theorem 1** ([4, 15]). *No algorithm for computing an $(1+\alpha)$-approximate k-means clustering runs in time polynomial of n, d and Δ (or polynomial of n, d and k) unless P=NP.*

This lower bound does not rule out the possibility that this problem can be solved in $O(nd + f(k, \Delta))$ time for an exponential function f of k and Δ. However, it seems hard to achieve this goal using the framework of Kumar et al. [14] and Ackermann et al. [1] as their algorithms (for the standard clustering problem) also run in $O(nd \cdot f(k))$ time for an exponential function f of k.

▶ Remark 2. Although it seems hard to achieve a significantly better running time using the framework of Ackermann et al., there is a merit of using this framework: we can handle outliers without additional effort as shown in [9]. Details will be discussed in Conclusion.

2 Preliminaries

We consider points in $\mathbb{R}^d$ with missing entries in some coordinates. Let us denote the missing entry value by $\otimes$, and let $\mathbb{H}^d$ denote the set of elements of $\mathbb{R}^d$ where we allow some coordinates to take the value $\otimes$. Furthermore, we call a point in $\mathbb{H}^d$ a Δ-*missing point* if at most Δ of its coordinates have value $\otimes$. We use $[k]$ to denote the set $\{1, \ldots, k\}$ for any integer $k \geq 1$. For any point $u \in \mathbb{H}^d$ and an index $i \in [d]$, we use $(u)_i$ to denote the entry of the i-th coordinate of u. If it is clear from the context, we simply use u_i to denote $(u)_i$. Throughout this paper, we use i or j to denote an index of the coordinates of a point, and t to denote an index of a sequence (of points or sets). We use $(u_t)_{t\in[k]}$ to denote a k-tuple consisting of $u_1, u_2, \ldots, u_k$.

Distance between two Δ-missing points. The *domain* of a point u in $\mathbb{H}^d$, denoted by $\textsc{dom}(u)$, is defined as the set of coordinate-indices $i \in [d]$ with $(u)_i \neq \otimes$. For a set I of coordinate-indices in $[d]$, we say that u is *fully defined* on I if $\textsc{dom}(u) \subseteq I$. Similarly, we say that u is *partially defined* on I if $\textsc{dom}(u) \cap I \neq \emptyset$. For a set P of points of $\mathbb{H}^d$ and a set I of coordinate-indices in $[d]$, we use $\mathsf{FD}(P, I)$ to denote the set of points of P fully defined on I. Similarly, we use $\mathsf{PD}(P, I)$ to denote the set of points of P partially defined on I. The *null* point is a point $p \in \mathbb{H}^d$ such that $(p)_i = \otimes$ for all indices $i \in [d]$. With a slight abuse of notation, we denote the null point by $\otimes$ if it is clear from the context. Also, we sometimes use I_t to denote $\textsc{dom}(u_t)$ if it is clear from the context.

Notice that a Δ-missing point in $\mathbb{H}^d$ can be considered as a Δ-dimensional affine subspace in $\mathbb{R}^d$. The distance between two Δ-missing points in $\mathbb{H}^d$ is defined as the Euclidean distance between their corresponding Δ-dimensional affine subspaces in $\mathbb{R}^d$. More generally, we define the *distance* between two points x and y in $\mathbb{H}^d$ *on* a set $I \subseteq [d]$ as

$$d_I(x, y) = \sqrt{\sum_{i\in I} |x_i - y_i|^2},$$

where $|a - b| = 0$ for $a = \otimes$ or $b = \otimes$ by convention.

The k-Means clustering of Δ-missing points. In this paper, we consider the *k-means clustering* of Δ-missing points of $\mathbb{H}^d$. As in the standard setting (for $\Delta = 0$), we wish to partition a given point set P into k clusters to minimize the sum of squared distances of each point to its cluster center. For any partition $(P_t)_{t\in[k]}$ of P into k clusters such that each cluster P_t is associated with a cluster center $c_t \in \mathbb{R}^d$, the *cost* of the partition is defined as the sum of squared distances of each point in P to its cluster center.

To be more precise, we define the *clustering cost* as follows. For a set $P \subset \mathbb{H}^d$ and a Δ-missing point y, we use $\textsc{cost}(P, y)$ to denote the sum of squared distances of each point in P to y. We also define the cost on a coordinate set $I \subseteq [d]$, denoted by $\textsc{cost}_I(P, y)$, as

the sum of squared distances on I between the points in P and their cluster centers. That is, $\sum_{x\in P} d_I(x,y)^2$. For convention, $\textsc{cost}_i(P,y) = \textsc{cost}_{\{i\}}(P,y)$ for $i \in [d]$. The clustering cost $\textsc{cost}((P_t)_{t\in[k]},(c_t)_{t\in[k]})$ of clustering $((P_t)_{t\in[k]},(c_t)_{t\in[k]})$ is defined as $\sum_{t\in[k]} \textsc{cost}(P_t,c_t)$.

Now we introduce two properties of an optimal clustering $((P_t^*)_{t\in[k]},(c_t^*)_{t\in[k]})$ that minimizes the clustering cost, which will be frequently used throughout this paper. For each cluster P_t^*, $\textsc{cost}(P_t^*,c_t)$ is minimized when c_t is the *centroid* of P_t^* [7]. That is, c_t^* is the centroid of P_t^*. For a set P of points in $\mathbb{H}^d$, the *centroid of* P, denoted by $c(P)$, is defined as

$$(c(P))_i = \begin{cases} \otimes & \text{if } \mathsf{PD}(P,i) = \emptyset, \\ \frac{\sum_{u\in\mathsf{PD}(P,i)} u_i}{|\mathsf{PD}(P,i)|} & \text{otherwise.} \end{cases}$$

Also, the clustering cost is minimized when $(P_t^*)_{t\in[k]}$ forms the Voronoi partition of P induced by $(c_t^*)_{t\in[k]}$ [7]. That is, $(P_t^*)_{t\in[k]}$ is the partition of P into k clusters in such a way that c_t^* is the closest cluster point from any point p in P_t^*.

Sampling. Our algorithm uses random sampling to compute an approximate k-means clustering. Lemma 3 is a restatement of [1, Lemma 2.1], and Lemma 4 is used in [7] implicitly. Since Lemma 4 is not explicitly stated in [7], we give a sketch of the proof in the full version of this paper [6].

▶ **Lemma 3** ([1, Lemma 2.1]). *Assume that we are given a set P of points in $\mathbb{H}^d$, an index $i \in [d]$, and an approximation factor $\alpha > 0$. Let Q be a subset of P with $|\mathsf{PD}(Q,i)| \geq c|P|$ for some constant c, which is not given explicitly. Then we can compute a point x of $\mathbb{R}$ in $O(|P|dm_{\alpha,\delta})$ time satisfying with probability $\frac{1-\delta}{5}2^{\Omega(-m_{\alpha,\delta}\log(\frac{1}{c}m_{\alpha,\delta}))}$ that*

$$\textsc{cost}_i(Q,x) \leq (1+\alpha)\textsc{cost}_i(Q,c(Q)),$$

where $m_{\alpha,\delta} \in O(1/(\alpha\delta))$.

▶ **Lemma 4** ([7]). *Assume that we are given a set P of Δ-missing points in $\mathbb{H}^d$ and an approximation factor $\alpha > 0$. Let Q be a subset of P with $|Q| \geq c|P|$ for some constant c with $0 < c < 1$, which is not given explicitly. Then we can compute a Δ-missing point $u \in \mathbb{H}^d$ in $O(|P|d\lambda)$ time satisfying with probability $\frac{c^{8(\Delta+1)\lambda+1}}{4(4\Delta)^{8\Delta\lambda}}$ that*

$$\textsc{cost}_I(Q,u) < (1+\alpha)\textsc{cost}_I(Q,c(Q)),$$

where I denotes the domain of u, and $\lambda = \max\{(\frac{3}{\alpha})^{1/(2\Delta)}, (128\Delta^3)^{1/(2\Delta)}\}$.

3 Overview of the Algorithm

We first briefly describe a $(1+\epsilon)$-approximation algorithm for k-means clustering for points in d-dimensional Euclidean space given by Kumar et al. [14]. Let P be a set of n points in $\mathbb{R}^d$, and $((P_t^*)_{t\in[k]},(c_t^*)_{t\in[k]})$ be an optimal k-means clustering for P.

Sketches of [1] and [14]. The algorithm of Kumar et al. [14] consists of several phases of two types: sampling phases and pruning phases. Their idealized strategy is as follows. At the beginning of a phase, it decides the type of the phase by computing the index t that maximizes $|P_t^*|$. If the cluster center of P_t^* has not been obtained, the algorithm enters the sampling phase. This algorithm picks a random sample of a constant size from P, and hopefully this sample would contain enough random samples from P_t^*. Then one can compute a good approximation c_t to c_t^* using Lemma 3.

If it is not the case, the algorithm enters a pruning phase, and it assigns each point to its closest cluster if their distance is at most L, where L denotes the smallest distance between two cluster centers we have obtained so far. They repeat this until all cluster centers are obtained, and finally obtain a good approximation to $(P_t^*)_{t\in[k]}$.

However, obviously, it is hard to implement this idealized strategy. To handle this, they try all possibilities (for both pruning and sampling phases and for all indices $t \in [k]$ to be updated for sampling phases), and return the best solution found this way. Kumar et al. [14] showed that their algorithm runs in $O(2^{(k/\epsilon)^{O(1)}}dn)$ time, and returns an $(1+\epsilon)$-approximate k-means clustering with probability $1/2$. Later, Ackermann et al. [1] gave a tighter bound on the running time of this algorithm.

Sketch of Eiben et al. [7]. To handle Δ-missing points, Eiben et al. generalized the algorithm in [14]. Their idealized strategy can be summarized as follows. It maintains k centers $(u_t)_{t\in[k]}$, which are initially set to the null points. In each sampling phase, it obtains one (or at least $[d]-\Delta$) coordinate of one of the centers.

At the beginning of a phase, it decides the type of the phase by computing the index t that maximizes $|\mathsf{PD}(P_t^*, [d]-I_t)|$. A sampling phase happens if $|\mathsf{PD}(P_t^*, [d]-I_t)| > c|R|$, where R denotes the number of points which are not yet assigned to any cluster. In this case, a random sample of constant size from R would contain enough random samples from $|\mathsf{PD}(P_t^*, j)|$ with $j \in [d]-I_t$. Thus, using the random sample, one can obtain a good approximation to $(c_t^*)_j$.

Otherwise, a pruning phase happens. In a pruning phase, the algorithm assigns points which are not yet assigned to any cluster to clusters. Here, a main difficulty is that even though the distance between a point p in R and its closest center u_t is at most L, where L denotes the distance between two cluster centers, p is not necessarily in P_t^*. They resolved this in a clever way by ignoring Δ coordinates for comparing the distances from two cluster centers.

Comparison of our contribution and Eiben et al. [7]. Our contribution is two-fold: the dependency on n decreases to $O(n)$ from $O(n^2)$, and the dependency of Δ and k decreases significantly. First, the improvement on the dependency of n comes from introducing a faster and simpler procedure for a pruning phase. In the previous algorithm, it cannot be guaranteed that a constant fraction of points of R is removed from R. This yields the quadratic dependency of n in their running time. We overcome the difficulty they (and we) face in a pruning phase in a different way. For each subset T of $[k]$, we consider the set S_T of points $x \in R$ such that $\textsc{dom}(x) \subset I_t$ for every $t \in T$ and $\textsc{dom}(x) \not\subset I_{t'}$ for every $t' \notin T$. Then S_T's for all subsets $T \subset [k]$ form a partition of R. In a pruning phase, we choose the set S_T that maximizes $|S_T|$. We show that the size of S_T is at least a constant fraction of $|R|$ (unless we enter the sampling phase). Moreover, in this case, if the distance between a point p in S_T and its closest center u_t is at most L, where L denotes the distance between two cluster centers, it holds that $p \in P_t^*$.

Second, we improved the dependency of Δ and k using the framework of Ackermann et al. [1]. To adapt this framework for our problem, we are required to handle several technical difficulties. This is mainly because the center of each cluster changes during the execution of the algorithm in our case unlike the standard setting considered in [1].

Due to the lack of space, some proofs are omitted. All missing proofs can be found in the full version of this paper.

Algorithm 1 **Idealized k-Means.**

input : A set P of Δ-missing points in the plane
output : A $(1+\epsilon)$-approximate k-means clustering for P

$R \leftarrow P$ and $P_t \leftarrow \emptyset$ for all cluster-indices $t \in [k]$
Initialize $\mathcal{U} = \langle u_1, \ldots, u_k \rangle$ so that $(u_t)_i = \otimes$ for all cluster-indices $t \in [k]$ and $i \in [d]$
while $R \neq \emptyset$ **do**
  Let t be the cluster-index that maximizes $|\mathsf{PD}(R \cap P_t^*, [d] - I_t)|$
  if $|\mathsf{PD}(R \cap P_t^*, [d] - I_t)| \geq c|R|$ **then**
    `/* sampling phase */`
    if $I_t = \emptyset$ **then**
      `/* We respresent this sampling phase as` $(t, \textsc{dom}(u_t))$ `*/`
      $u_t \leftarrow$ The Δ-missing point obtained from Lemma 4
    else
      `/* We respresent this sampling phase as` $(t, \{j\})$ `*/`
      Let j be the coordinate-index in $[d] - I_t$ that maximizes $\mathsf{PD}(R \cap P_t^*, j)$
      $(u_t)_j \leftarrow$ The value obtained from Lemma 3
    Assign the points in $\mathsf{FD}(R, \cap_{t \in [k]} I_t)$ to their closest cluster centers in $\mathcal{U}$
    $R \leftarrow R - \mathsf{FD}(R, \cap_{t \in [k]} I_t)$
  else
    `/* pruning phase */`
    `/*` $\mathcal{U}_T$ `is the set of all` u_t `for` $t \in T$ `*/`
    $T \leftarrow$ The set of cluster-indices that maximizes $|S_T|$
    $B \leftarrow$ The first half of S_T sorted in ascending order of distance from $\mathcal{U}_T$
    Assign the points in B to their closest cluster centers in $\mathcal{U}_T$
    $R \leftarrow R - B$
return $\mathcal{U}$

4 For k-Means Clustering

In this section, we describe and analyze for a k-means clustering algorithm. Let $\mathcal{U} = \langle u_1, u_2, \ldots, u_k \rangle$ be a sequence of points in $\mathbb{H}^d$.

4.1 Algorithm Using the Counting Oracle

We first sketch an algorithm for k-clustering assuming that we can access the *counting oracle*. Let $\langle P_1^*, \cdots, P_k^* \rangle$ be an optimal k-clustering for P induced by the centroids $\langle c_1^*, \cdots, c_k^* \rangle$. The counting oracle takes a subset of P and a cluster-index $t \in [k]$, and it returns the number of points in the subset which are contained in P_t^*. Then in Section 4.3, we show how to compute an approximate clustering without the counting oracle.

The algorithm consists of several phases of two types: a *sampling phase* or a *pruning phase*. We initialize $\mathcal{U}$ to a k-tuple of null points. In a sampling phase, we obtain values of $(u_t)_j$ for indices $t \in [k]$ and $j \in [d]$ which were set to $\otimes$. Also, we assign points of P to one of the k clusters in sampling and pruning phases. The pseudocode of the algorithm is described in Algorithm 1. At any moment during the execution of the algorithm, we maintain a set R of remaining points and a k-tuple $\mathcal{U} = \langle u_1, \ldots, u_k \rangle$ of (partial) centers in $\mathbb{H}^d$. We let I_t be $\textsc{dom}(u_t)$. Initially, R is set to P, and $\mathcal{U}$ is set to the k-tuple of null points. The algorithm terminates if $R = \emptyset$, and finally $\mathcal{U}$ becomes a set of points in $\mathbb{R}^d$.

We consider the partition $\mathcal{F}$ of R defined as follows. For a subset T of $[k]$, let S_T denote the set of points $x \in R$ such that $\textsc{dom}(x) \subset I_t$ for every $t \in T$ and $\textsc{dom}(x) \not\subset I_{t'}$ for every $t' \notin T$. Let $\mathcal{F} = \{S_T \mid T \text{ is a proper subset of } [k]\}$. The following lemma shows that $\mathcal{F}$ is a partition of R.

▶ **Lemma 5.** *For any point $x \in R$, there exists a unique set in $\mathcal{F}$ containing x.*

At the beginning of each phase, we decide the type of the current phase. Let t be the cluster-index of $[k]$ that maximizes $|\mathsf{PD}(R \cap P_t^*, [d] - I_t)|$, where R is the set of points of P which are not assinged to any cluster. The one of the following cases always happen: $|\mathsf{PD}(R \cap P_t^*, [d] - I_t)| \geq c|R|$, or there exists a set T of cluster-indices in $[k]$ such that $|S_T \cap (\cup_{t \in T} P_t^*)| \geq c|R|$ for any constant $c < 1/(2^k + k)$, which will be specified later.[1]

▶ **Lemma 6.** *One of the following always holds for any constant $c < 1/(2^k + k)$.*
- *$|\mathsf{PD}(R \cap P_t^*, [d] - I_t)| \geq c|R|$ for a cluster-index $t \in [k]$, or*
- *$|S_T \cap (\cup_{t \in T} P_t^*)| \geq c|R|$ for a proper subset T of $[k]$.*

Sampling phase. If the first case happens, we enter a sampling phase. Let α be a constant, which will be specified later. We use it as an approximation factor used in Lemmas 3 and 4 for sampling. If I_t is empty, we replace u_t with a Δ-missing point in $\mathbb{H}^d$ obtained from Lemma 4. If I_t is not empty, then it is guaranteed that $|I_t|$ is at least $d - \Delta$. We compute the coordinate-index j in $[d] - I_t$ that maximizes $|\mathsf{PD}(R \cap P_t^*, j)|$ using the counting oracle. Clearly, $(u_t)_j = \otimes$ and $|\mathsf{PD}(R \cap P_t^*, j)|$ is at least $c|R|/\Delta$. Then we replace $(u_t)_j$ with a value obtained from Lemma 3. At the end of the phase, we check if $\mathsf{FD}(R, \cap_{t \in [k]} I_t)$ is not empty. If it is not empty, we assign those points to their closest cluster centers.

Pruning phase. Otherwise, we enter a pruning phase. Instead of obtaining a new coordinate value of u_t, we assign points of R to cluster centers in a pruning phase. To do this, we find a proper subset T of $[k]$ which maximizes $|S_T|$. Then among the points of S_T, we choose the $|S_T|/2$ points closest to their closest centers in $\mathcal{U}_T$, where $\mathcal{U}_T$ is the set of all u_t for $t \in T$. Then we assign each of them to its closest center in $\mathcal{U}_T$. In this way, points in $\cup_{t' \notin T} P_{t'}^*$ might be assigned (incorrectly) to u_t for a cluster-index $t \in T$. We call such a point a *stray point*.

4.2 Analysis of the Approximation Factor

In this section, we analyze the approximation factor of the algorithm. We let $\mathcal{S}$ be the sequence of sampling phases happened during the execution of the algorithm. At the end of the algorithm, $\mathcal{U}$ becomes a $(1+\epsilon)$-approximate clustering as we will see later. In the following, we use $\mathcal{C} = \langle c_1, \ldots, c_k \rangle$ to denote the output of the algorithm. For a sequence T of cluster-indices, we use $\mathcal{C}_T$ to denote a $|T|$-tuple consisting of c_t for $t \in T$. Let $\textsc{opt}_k(P)$ denote the clustering cost of an optimal k-means clustering of P.

▶ **Lemma 7.** *The size of $\mathcal{S}$ is at most $k(\Delta + 1)$.*

Preliminaries. Recall that $\mathcal{S}$ denotes the sequence of sampling phases. Here, we represent a sampling phase as the pair (t, I), where t is the cluster-index considered in the sampling phase, and I is the set of indices i such that $(u_t)_i$ is obtained during the sampling phase.

[1] We set $\alpha = \epsilon/3$, and $c = \frac{\alpha}{8 \cdot 2^k k^2 (\Delta+1)}$.

Note that the size of I is either one or at least $d - \Delta$ for any sampling phase. Also, for $s = (t, I) \in \mathcal{S}$, we let $t^s = t$ and $I^s = I$. For each sampling phase $s = (t, I)$, let R^s be the set of points of P_t^* which are not assigned at the *beginning* of the phase. Furthermore, let $I^s_{t'}$ denote $\textsc{dom}(u_{t'})$ we have at the end of the sampling phase s for a cluster-index $t' \in [k]$. For two sampling phases s and s' in $\mathcal{S}$, we use $s \preceq s'$ if s comes before s' in $\mathcal{S}$ or equals to s'. We denote $P^*_T = \cup_{t \in T} P^*_t$ and $P^*_{\bar{T}} = \cup_{t' \notin T} P^*_{t'}$.

Sketch. A point of P is assigned to one of the clusters during a sampling phase or a pruning phase. That is, at the end of the execution of the algorithm, P is partitioned into k clusters $P_1, \ldots, P_k$. Note that it is not necessarily a Voronoi partition of P with respect to $\mathcal{C}$. Our goal in this section is that the clustering cost $\textsc{cost}((P_t)_{t\in[k]}, (c_t)_{t\in[k]})$ is at most $(1+\epsilon)\textsc{opt}_k(P)$. We first show that the clustering cost induced by non-stray points is $(1+\alpha)\textsc{opt}_k(P)$, and then show that the clustering cost induced by stray points is $\alpha\textsc{opt}_k(P)$.

For this, we use the two following technical lemmas. Lemma 8 is a consequence of Lemmas 3 and 4, and Claim 9 follows from construction. Proofs can be found in the Appendix of the full version [6].

▶ **Lemma 8.** *$\sum_{s\in\mathcal{S}} \textsc{cost}_{I^s}(R^s, c_{t^s}) \le (1+\alpha)\textsc{opt}_k(P)$ with a probability at least $p^k q^{k\Delta}$, where q and p are the probabilities in Lemmas 3 and 4.*

▷ Claim 9. For a sampling phase s' in $\mathcal{S}$ and a proper subset T of $[k]$, let X be a point subset of S_T which are not assigned at the end of a sampling phase s'. Then

$$\textsc{cost}(X \cap P^*_T, \mathcal{C}_T) \le \sum \textsc{cost}_{I^s}(\mathsf{PD}(X \cap P^*_{t^s}, I^s), c_{t^s}),$$

where the summation is taken over all sampling phases s in $\mathcal{S}$ with $s \preceq s'$ and $t^s \in T$.

4.2.1 Clustering Cost Induced by Non-Stray Points

We first show that the clustering cost induced by non-stray points is at most $(1+\alpha)\textsc{opt}_k(P)$. There are two types of non-stray points: the points assigned during the sampling phases, and the points assigned during the pruning phases which are not stray. The first term in the following lemma is the clustering cost induced by points of the first type, and the second term is the clustering cost induced by points of the second type. For a proper subset T of $[k]$, let A^s_T be the set of points in S_T assigned to $\mathcal{U}$ during the consecutive pruning phases lying between two adjacent sampling phases s and s' with $s \preceq s'$.

▶ **Lemma 10.** *Let S be the set of points assigned during the sampling phases.*

$$\textsc{cost}(S, \mathcal{C}) + \sum_{s\in\mathcal{S}} \sum_{T \subsetneq [k]} \textsc{cost}(A^s_T \cap P^*_T, \mathcal{C}_T) \le (1+\alpha)\textsc{opt}_k(P),$$

with a probability at least $p^k q^{k\Delta}$, where q and p are the probabilities in Lemmas 3 and 4.

Proof. A point $p \in S$ is assigned to its closest center in $\mathcal{C}$. This is because we assign a point during a sampling phase only if it is fully defined on $\textsc{dom}(u_t)$ for all cluster-indices $t \in [k]$. Thus the following holds.

$$\textsc{cost}(S, \mathcal{C}) \le \sum_{t\in[k]} \textsc{cost}(S \cap P^*_t, c_t) = \sum_{s\in\mathcal{S}} \textsc{cost}_{I^s}(\mathsf{PD}(S \cap R^s, I), c_{t^s}).$$

For the second term of this claim, we have the following:

$$\begin{aligned}\sum_{s'\in\mathcal{S}}\sum_{T\subsetneq[k]}\textsc{cost}(A_T^{s'}\cap P_T^*,\mathcal{C}_T) &\le \sum_{s'\in\mathcal{S}}\sum_{T\subsetneq[k]}\sum_{\substack{s\preceq s'\\ t^s\in T}}\textsc{cost}_{I^s}(\mathsf{PD}(A_T^{s'}\cap P_{t^s}^*,I^s),c_{t^s})\\ &= \sum_{s\in S}\sum_{s\preceq s'}\sum_{t^s\in T\subsetneq[k]}\textsc{cost}_{I^s}(\mathsf{PD}(A_T^{s'}\cap P_{t^s}^*,I^s),c_{t^s})\\ &\le \sum_{s\in S}\sum_{s\preceq s'}\sum_{t^s\in T\subsetneq[k]}\textsc{cost}_{I^s}(\mathsf{PD}(A_T^{s'}\cap R^s,I^s),c_{t^s})\\ &\le \sum_{s\in S}\textsc{cost}_{I^s}(\mathsf{PD}(A\cap R^s,I^s),c_{t^s}),\end{aligned}$$

where A denotes the set of points of P assigned during all pruning phases. The first inequality holds by Claim 9. The second and the last inequalities hold since they change only the ordering of summation. The third inequality holds since $A_T^{s'}\cap P_{t^s}^*\subset R^s$ if $s\preceq s'$.

By combining previous properties together with Claim 9, we have:

$$\begin{aligned}\textsc{cost}(S,\mathcal{C})+\sum_{s\in\mathcal{S}}\sum_{T\subsetneq[k]}\textsc{cost}(A_T^s\cap P_T^*,\mathcal{C}_T) &\le \sum_{s\in\mathcal{S}}\textsc{cost}_{I^s}(\mathsf{PD}(S\cap R^s,I^s),c_{t^s})\\ &\quad+\sum_{s\in S}\textsc{cost}_{I^s}(\mathsf{PD}(A\cap R^s,I^s),c_{t^s})\\ &\le \sum_{s\in\mathcal{S}}\textsc{cost}_{I^s}(R^s,c_{t^s})\\ &\le (1+\alpha)\textsc{opt}_k(P).\end{aligned}$$

◀

4.2.2 Clustering Cost Induced by Stray Points

Now we show that the clustering cost induced by stray points is $\alpha\textsc{opt}_k(P)$. Recall that the stray points are assigned to clusters during pruning phases. We first analyze the clustering cost by considering consecutive pruning phases lying between two adjacent sampling phases of $\mathcal{S}$. Then we show that the overall clustering cost induced by stray points.

4.2.2.1 During consecutive pruning phases

Consider a sequence $\mathcal{P}$ of the consecutive pruning phases lying between two adjacent sampling phases s and s' in $\mathcal{S}$ with $s\preceq s'$. Let N denote the number of pruning phases in $\mathcal{P}$. For a proper subset T of $[k]$, recall that A_T^s denotes the set of points in S_T assigned to $\mathcal{U}$ during the phases in $\mathcal{P}$. During this period, u_t remains the same for each cluster-index $t\in[k]$. By definition, $A_T^s\cap P_{\bar{T}}^*$ is the set of stray points assigned during the pruning phases in $\mathcal{P}$. For each pruning phase of $\mathcal{P}$, we choose a subset T of $[k]$ and assign a half of S_T to $\mathcal{U}$. Here, note that distinct pruning phases of $\mathcal{P}$ might choose distinct subsets T. Therefore, there might be more than one subsets T of $[k]$ with $A_T^s\neq\emptyset$.

We first analyze the clustering cost induced by stray points assigned during the pruning phases in $\mathcal{P}$. Recall that R^s denotes the set of points of P which are not yet assigned to any cluster at the end of the sampling phase s. Let S_T^s denotes the set of points of $x\in R^s$ such that $\textsc{dom}(x)\subset I_t$ for every $t\in T$ and $\textsc{dom}(x)\not\subset I_{t'}$ for every $t'\notin T$.

▶ **Lemma 11.** $\sum_{T\subsetneq[k]}\textsc{cost}(A_T^s\cap P_{\bar{T}}^*,\mathcal{C}_T)\le 8\cdot 2^k ck\cdot\sum_{T\subsetneq[k]}\textsc{cost}(S_T^s\cap P_T^*,\mathcal{C}_T)$.

Proof (Sketch). Since S_T^s's form a partition of R^s, it suffices to fix a proper subset T of $[k]$ and show that $\textsc{cost}(A_T^s \cap P_{\bar{T}}^*, \mathcal{C}_T) \leq 8 \cdot 2^k ck \cdot \textsc{cost}(S_T^s \cap P_T^*, \mathcal{C}_T)$. We partition A_T^s into N subsets so that $A_T^{(x)}$ is the set of points of S_T assigned to $\mathcal{U}$ at the xth pruning phase of $\mathcal{P}$. By construction, in the xth pruning phase, there exists a unique index-set T with $A_T^{(x)} \neq \emptyset$. Let $\mathcal{X}_T$ be the increasing sequence of indices x of $[N]$ with $A_T^{(x)} \neq \emptyset$.

For any consecutive indices x and x' in $\mathcal{X}_T$ with $x' < x$ and any proper subset T of $[k]$, we show that the clustering cost induced by stray points assigned during the x'th pruning phase of $\mathcal{P}$ is at most $8 \cdot 2^k ck$ times the clustering cost induced by non-stray points assigned during the xth pruning phase of $\mathcal{P}$. Also, we analyze the clustering cost induced by stray points assigned during the last phase of $\mathcal{P}$ similarly. The clustering cost induced by all non-stray points assigned during the pruning phases of $\mathcal{P}$ is at most $\sum_{T \subsetneq [k]} \textsc{cost}(S_T^s \cap P_T^*, \mathcal{C}_T)$, and thus the lemma holds. Details can be found in the full version of this paper. ◀

4.2.2.2 During the entire pruning phases

Recall that $\mathcal{S}$ denotes the sequence of sampling phases, and each sampling phase is represented as (t, I), where t is the cluster-index considered in the sampling phase, and I is the set of indices i such that $(u_t)_i$ is obtained during the sampling phase.

The following lemma gives an upper bound of the total cost induced by the stray points.

▶ **Lemma 12.** $\displaystyle\sum_{s \in \mathcal{S}} \sum_{T \subsetneq [k]} \textsc{cost}(A_T^s \cap P_{\bar{T}}^*, \mathcal{C}_T) \leq 8 \cdot 2^k ck^2 (\Delta + 1)(1 + \alpha) \textsc{opt}_k(P)$,

with a probability at least $p^k q^{k\Delta}$, where q and p are the probabilities in Lemmas 3 and 4.

Proof. The lemma holds by the following inequalities. The first and second inequalities hold by Claims 11 and 9, respectively. The third one holds since it changes only the ordering of summation. The fourth one holds since for a fixed sampling phase s' in $\mathcal{S}$, $S_T^{s'}$ are disjoint for all proper subsets T of $[k]$. Also notice that, for two sampling phases s, s' in $\mathcal{S}$ and a proper subset T of $[k]$, R^s contains $S_T^{s'} \cap P_{t^s}^*$ if $s \preceq s'$. The fifth one holds since the size of $\mathcal{S}$ is at most $k(\Delta + 1)$. The last two hold by the definition of $\textsc{cost}(\cdot)$.

$$
\begin{aligned}
\sum_{s \in \mathcal{S}} \sum_{T \subsetneq [k]} \textsc{cost}(A_T^s \cap P_{\bar{T}}^*, \mathcal{C}_T) &\leq 8 \cdot 2^k ck \sum_{s \in \mathcal{S}} \sum_{T \subsetneq [k]} \textsc{cost}(S_T^s \cap P_T^*, \mathcal{C}_T) \\
&\leq 8 \cdot 2^k ck \sum_{s' \in \mathcal{S}} \sum_{T \subsetneq [k]} \sum_{\substack{s \preceq s' \\ t^s \in T}} \textsc{cost}_{I^s}(\mathsf{PD}(S_T^{s'} \cap P_{t^s}^*, I^s), c_{t^s}) \\
&= 8 \cdot 2^k ck \sum_{s \in S} \sum_{s \preceq s'} \sum_{t^s \in T \subsetneq [k]} \textsc{cost}_{I^s}(\mathsf{PD}(S_T^{s'} \cap P_{t^s}^*, I^s), c_{t^s}) \\
&\leq 8 \cdot 2^k ck \sum_{s \in \mathcal{S}} \sum_{s \prec s'} \textsc{cost}_{I^s}(\mathsf{PD}(R^s, I^s), c_{t^s}) \\
&\leq 8 \cdot 2^k ck^2 (\Delta + 1) \sum_{s \in \mathcal{S}} \textsc{cost}_{I^s}(\mathsf{PD}(R^s, I^s), c_{t^s}) \\
&\leq 8 \cdot 2^k ck^2 (\Delta + 1) \sum_{s \in \mathcal{S}} \textsc{cost}_{I^s}(R^s, c_{t^s}), \\
&\leq 8 \cdot 2^k ck^2 (\Delta + 1)(1 + \alpha) \textsc{opt}_k(P).
\end{aligned}
$$

◀

We can obtain the following lemma by combining Lemma 10 and Lemma 12.

▶ **Lemma 13.** *For a constant $\alpha > 0$, the algorithm returns an $(1 + 8 \cdot 2^k ck^2(\Delta + 1))(1 + \alpha)$-approximate k-means clustering for P with probability at least $p^k q^{k\Delta}$, where q and p are the probabilities in Lemmas 3 and 4.*

Algorithm 2 k-Means.

```
R ← R − FD(R, ∩_{t∈[k]} I_t)
E ← ∅
U' = ⟨u'_1, ..., u'_k⟩ ← U = ⟨u_1, ..., u_k⟩
if R = ∅ then return u_[k]
for t ∈ [k] do
    if I_t = ∅ then
        u'_t ← the Δ-missing point obtained from Lemma 4
        Add the clustering returned by k-Means (U', R) to E
    else
        foreach j ∈ [d] − I_t do
            u'_t ← u_t
            (u'_t)_j ← The value obtained from Lemma 3
            Add the clustering returned by k-Means (U', R) to E
T ← the non-empty proper subset of [k] that maximizes |R ∩ S_T|
if |R ∩ S_T| ≥ |R|/(2^k − 1) then
    /* U_T is the set of all u_t for t ∈ T */
    B ← The first half of |R ∩ S_T| sorted in ascending order of distance from U_T
    Add the clustering returned by k-Means (U, R − B) to E
return the clustering C in E which minimizes cost(R, C)
```

4.3 Algorithm without Counting Oracle

The algorithm we have described uses the counting oracle in two places: determining the type of the phase and selecting a pair of the cluster-index and coordinate-index to be updated in a sampling phase. In this section, we explain how to avoid using the counting oracle. To do this, we simply try all possible cases: run both phases and update each possible cluster for all indices during a sampling phase. The main algorhm, k-Means $(\mathcal{U}, R)$, is described in Algorithm 2. Its input consists of cluster centers $\mathcal{U}$ of a partial clustering of P and a set R of points of P which are not yet assigned. Finally, k-Means $(\otimes_k, P)$ returns an $(1 + 8 \cdot 2^k ck^2(\Delta + 1))(1 + \alpha)$-approximate k-means clustering of P, where $\otimes_k$ denotes the k-tuple of $\mathbb{H}^d$ of null points.

The clustering cost returned by k-Means $(\otimes_k, P)$ is at most the cost returned by the algorithm which uses the counting oracle in Section 4.1. In the following, we analyze the running time of k-Means $(\otimes_k, P)$. Let $T(n, \delta)$ be the running of k-Means $(\mathcal{U}, R)$ when $n = |R|$ and $\delta = \sum_{t\in[k]} \min\{d - |I_t|, \Delta + 1\}$. Here, δ is an upper bound on the number of updates required to make $I_t = [d]$ for every cluster-index t in $[k]$. Then we have the following recurrence relation for $T(n, \delta)$.

▷ Claim 14. $T(n, \delta) \le \delta \cdot T(n, \delta - 1) + T\left(\left(1 - \frac{1}{2^{k+1}-2}\right) n, \delta\right) + O(\frac{k\delta\Delta^3 dn}{\alpha})$

Proof. In a sampling phase, k-Means calls itself at most δ times recursively with different parameters. Each recursive call takes $T(n, \delta - 1)$ time. Also, the time for updating cluster centers takes $O(\delta\Delta^3 dn/\alpha)$ in total by Lemma 3 and 4. For a pruning phase, we compute $|R \cap S_T|$ for each $T \subsetneq [k]$ in total $O(dn)$ time, and then choose the first half of S_T in increasing order of the distances from u_T in total $O(kdn)$ time. The recursive call invoked in the pruning phase takes $T\left(\left(1 - \frac{1}{2^{k+1}-2}\right) n, \delta\right)$ time. We have $\delta + 1$ clusterings returned by recursive calls in total, and we can choose $c_{[k]}$ in $O(\delta kdn)$ time. Thus, the claim holds. ◁

By solving the recurrence relation, we can obtain an upper bound of $T(n, \delta)$.

▷ Claim 15. $T(n, \delta) \leq (2\delta(2^k - 1))^{2\delta+1}(1 + \frac{1}{2^{k+1}-3})^{\delta^2}\Delta^3 kdn/\alpha$

We obtain the following theorem by setting $\alpha = \epsilon/3$, and $c = \frac{\alpha}{8 \cdot 2^k k^2(\Delta+1)}$.

▶ **Theorem 16.** *Given a Δ-missing n-point set P in $\mathbb{H}^d$, a $(1+\epsilon)$-approximate solution to the k-means clustering problem can be found in $2^{O(\max\{\Delta^4 k(\log \Delta + k),\ \frac{\Delta k}{\epsilon}(\log \frac{1}{\epsilon} + k)\})}dn$ time with a constant probability $1/2$.*

5 Concluding Remarks

In this paper, we gave a linear-time approximation algorithm for k-means clustering on axis-parallel affine subspaces. Our algorithm runs in time linear in nd, which is the size of the input. The bound is almost tight in the sense that no $(1+\epsilon)$-approximation algorithm for this problem runs in time polynomial in even one of k and Δ.

Our algorithm is based on the framework of Kumar et al. [14] and Ackerman et al. [1]. One merit of using this framework is that we can handle outliers without additional effort as shown in [9]. In this case, the goal is to minimize the clustering cost allowing to remove a small portion of the input data. The problem of computing a k-means clustering of missing data has not been explicitly considered in the presence of outliers. However, the observation of [9] allows us to extend our algorithm to handle outliers.

A main idea of [9] for clustering points in $\mathbb{R}^d$ is as follows. Let m be the number of outliers (the number of points which are allowed to be removed). Consider an optimal k-means clustering $(P_1^*, \ldots, P_k^*)$ of the input point set P after removing m outliers. Then the largest cluster, say P_1^*, has size at least $(|P| - m)/k$. Then by picking a random sample of a constant size (but the sample size depends on m), one can compute a good approximation to $c(P_1^*)$ using Lemma 3. Using this observation, Feng et al. [9] showed that the algorithm by Kumar et al. [14] (using a parameter slightly larger than the standard one for Lemma 3) computes the cluster centers of an approximation k-means clustering in the presence of m outliers. Then the m points farthest from the cluster centers are m outliers.

The observation by Feng et al. also holds for Δ-missing points. In a sampling phase, the set R of remaining points contains at most m outliers. This means that the largest set S_T contains at least $|R|/(2^k + k + m)$ points of P_T^*. Therefore, we can apply Lemma 6 using a constant $c < 1/(2^k + k + m)$. Thus we can handle m outliers in $O(nd)$ time, where the constant hidden behind $O(\cdot)$ depends on m.

As mentioned in Introduction, the lower bound in Theorem 1 does not rule out the possibility that this problem can be solved in $O(nd + f(k, \Delta))$ time for an exponential function f of k and Δ. Moreover, it seems hard to achieve this goal using the framework of Kumar et al. [14] and Ackermann et al. [1] as their algorithms also run in $O(nd \cdot f(k))$ time for an exponential function of k. It is an interesting open question whether one can improve the running time to $O(nd + f(k, \Delta))$. Also, obtaining a coreset for this problem is also an interesting open question.

References

1 Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Transactions on Algorithms*, 6(4), September 2010.

2 P.D. Allison. *Missing Data*. Number no. 136 in Missing Data. SAGE Publications, 2001. URL: `https://books.google.co.kr/books?id=ZtYArHXjpB8C`.

3 Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.

4 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of Euclidean k-means. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG 2015)*, 2015.

5 Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

6 Kyungjin Cho and Eunjin Oh. Linear-time approximation scheme for k-means clustering of affine subspaces. *CoRR*, abs/2106.14176, 2021. `arXiv:2106.14176`.

7 Eduard Eiben, Fedor V Fomin, Petr A Golovach, Willian Lochet, Fahad Panolan, and Kirill Simonov. EPTAS for k-means clustering of affine subspaces. In *Proceedings of the Thirty-Second ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 2649–2659, 2021.

8 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43th Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 569–578, 2011.

9 Qilong Feng, Zhen Zhang, Ziyun Huang, Jinhui Xu, and Jianxin Wang. Improved algorithms for clustering with outliers. In *Proceedings of the 30th International Symposium on Algorithms and Computation (ISAAC 2019)*, pages 61:1–61:12, 2019.

10 Jie Gao, Michael Langberg, and Leonard J Schulman. Analysis of incomplete data and an intrinsic-dimension helly theorem. *Discrete & Computational Geometry*, 40(4):537–560, 2008.

11 Jie Gao, Michael Langberg, and Leonard J. Schulman. Clustering lines in high-dimensional space: Classification of incomplete data. *ACM Trans. Algorithms*, 7(1), 2010.

12 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, January 2007.

13 Anil Kumar Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

14 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *Journal of the ACM*, 57(2):1–32, 2010.

15 Euiwoong Lee and Leonard J Schulman. Clustering affine subspaces: hardness and algorithms. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms (SODA 2013)*, pages 810–827, 2013.

16 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012.

17 Yair Marom and Dan Feldman. k-means clustering of lines for big data. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

18 Nimrod Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10(3):327–334, 1990.

19 Björn Ommer and Jitendra Malik. Multi-scale object detection by clustering lines. In *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV 2009)*, pages 484–491, 2009.