Approximate Clustering with Same-Cluster Queries*

Nir Ailon[†1], Anup Bhattacharya[‡2], Ragesh Jaiswal[3], and Amit Kumar[4]

1 **Technion, Haifa, Israel**
nailon@cs.technion.ac.il
2 **Department of Computer Science and Engineering, Indian Institute of Technology Delhi India**
anupb@cse.iitd.ac.in
3 **Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India**
rjaiswal@cse.iitd.ac.in
4 **Department of Computer Science and Engineering, Indian Institute of Technology Delhi, India**
amitk@cse.iitd.ac.in

Abstract

Ashtiani *et al.* proposed a Semi-Supervised Active Clustering framework (SSAC), where the learner is allowed to make adaptive queries to a domain expert. The queries are of the kind "*do two given points belong to the same optimal cluster?*", where the answers to these queries are assumed to be consistent with a unique optimal solution. There are many clustering contexts where such *same cluster* queries are feasible. Ashtiani *et al.* exhibited the power of such queries by showing that any instance of the k-means clustering problem, with additional *margin* assumption, can be solved efficiently if one is allowed to make $O(k^2 \log k + k \log n)$ same-cluster queries. This is interesting since the k-means problem, even with the margin assumption, is NP-hard.

In this paper, we extend the work of Ashtiani *et al.* to the approximation setting by showing that a few of such same-cluster queries enables one to get a polynomial-time $(1+\varepsilon)$-approximation algorithm for the k-means problem without any margin assumption on the input dataset. Again, this is interesting since the k-means problem is NP-hard to approximate within a factor $(1+c)$ for a fixed constant $0 < c < 1$. The number of same-cluster queries used by the algorithm is $\text{poly}(k/\varepsilon)$ which is independent of the size n of the dataset. Our algorithm is based on the D^2-sampling technique, also known as the k-means++ seeding algorithm. We also give a conditional lower bound on the number of same-cluster queries showing that if the Exponential Time Hypothesis (ETH) holds, then any such efficient query algorithm needs to make $\Omega\left(\frac{k}{\text{poly}\log k}\right)$ same-cluster queries. Our algorithm can be extended for the case where the query answers are wrong with some bounded probability. Another result we show for the k-means++ seeding is that a small modification of the k-means++ seeding within the SSAC framework converts it to a constant factor approximation algorithm instead of the well known $O(\log k)$-approximation algorithm.

1998 ACM Subject Classification I.5.3 Clustering

Keywords and phrases k-means, semi-supervised learning, query bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.40

* A full version of the paper is available at [3], https://arxiv.org/abs/1704.01862

† Nir Ailon and Ragesh Jaiswal acknowledge the support of ISF-UGC India-Israel Joint Research Grant 2014.

‡ Anup Bhattacharya acknowledges the support of TCS graduate fellowship at IIT Delhi.

1 Introduction

Clustering is extensively used in data mining and is typically the first task performed when trying to understand large data. Clustering basically involves partitioning given data into groups or clusters such that data points within the same cluster are similar as per some similarity measure. Clustering is usually performed in an unsupervised setting where data points do not have any labels associated with them. The partitioning is done using some measure of similarity/dissimilarity between data elements. In this paper, we extend the work of Ashtiani *et al.* [6] who explored the possibility of performing clustering in a *semi-supervised active learning* setting for center based clustering problems such as k-median/k-means. In this setting, which they call Semi-Supervised Active Clustering framework or SSAC in short, the clustering algorithm is allowed to make adaptive queries of the form: " *do two points from the dataset belong to the same optimal cluster?*", where query answers are assumed to be consistent with a unique optimal solution. Ashtiani *et al.* [6] started the study of understanding the strength of this model. Do hard clustering problems become easy in this model? They explored such questions in the context of center-based clustering problems. Center based clustering problems such as k-means are extensively used to analyze large data clustering problems. Let us define the k-means problem in the Euclidean setting.

▶ **Definition 1** (k-means problem)**.** Given a dataset $X \subseteq \mathbb{R}^d$ containing n points, and a positive integer k, find a set of k points $C \subseteq \mathbb{R}^d$ (called centers) such that the following cost function is minimized: $\Phi(C, X) = \sum_{x \in X} \min_{c \in C} D(x, c)$, where $D(x, c)$ denotes the squared Euclidean distance between x and c. That is, $D(x, c) = ||x - c||^2$.

Note that the k optimal centers $c_1, \ldots, c_k$ of the k-means problem define k clusters of points in a natural manner. All points for which the closest center is c_i belong to the i^{th} cluster. This is also known as the *Voronoi partitioning* and the clusters obtained in this manner using the optimal k centers are called the optimal clusters. Note that the optimal center for the 1-means problem for any dataset $X \subseteq \mathbb{R}^d$ is the centroid of the dataset denoted by $\mu(X) \stackrel{def.}{=} \frac{\sum_{x \in X} x}{|X|}$. This means that if $X_1, ..., X_k$ are the optimal clusters for the k-means problem on any dataset $X \subseteq \mathbb{R}^d$ and $c_1, ..., c_k$ are the corresponding optimal centers, then $\forall i, c_i = \mu(X_i)$. The k-means problem has been widely studied and various facts are known about this problem. The problem is tractable when either the number k of clusters or the dimension d is equal to 1. However, when $k > 1$ or $d > 1$, then the problem is known to be NP-hard [16, 32, 28]. There has been a number of works of *beyond the worst-case* flavour for the k-means problem in which it is typically assumed that the dataset satisfies some *separation* condition, and then the question is whether this assumption can be exploited to design algorithms with better guarantees for the problem. Such studies have led to different definitions of separation and also some interesting results for datasets that satisfy these separation conditions (e.g., [30, 11, 8]). Ashtiani *et al.* [6] explored one such separation notion that they call the γ-margin property.

▶ **Definition 2** (γ-margin property)**.** Let $\gamma > 1$ be a real number. Let $X \subseteq \mathbb{R}^d$ be any dataset and k be any positive integer. Let $P_X = \{X_1, \ldots, X_k\}$ denote k optimal clusters for the k-means problem. Then this optimal partition of the dataset P_X is said to satisfy the γ-margin property iff for all $i \neq j \in \{1, \ldots, k\}$ and $x \in X_i$ and $y \in X_j$, we have:

$$\gamma \cdot ||x - \mu(X_i)|| < ||y - \mu(X_i)||.$$

Qualitatively this means that every point within some cluster is closer to its own cluster center than a point that does not belong to this cluster. This seems to be a very strong separation

property. Ashtiani *et al.* [6] showed that the k-means clustering problem is NP-hard even when restricted to instances that satisfy the γ-margin property for $\gamma = \sqrt{3.4} \approx 1.84$. Here is the formal statement of their hardness result.

▶ **Theorem 3** (Theorem 10 in [6]). *Finding an optimal solution to the k-means objective function is NP-hard when $k = \Theta(n^{\varepsilon})$ for any $\varepsilon \in (0, 1)$, even when there is an optimal clustering that satisfies the γ-margin property for $\gamma = \sqrt{3.4}$.*

In the context of the k-means problem, the *same-cluster* queries within the SSAC framework are decision questions of the form: *Do points x, y such that $x \neq y$ belong to the same optimal cluster?* [1] Following is the main question explored by Ashtiani *et al.* [6]: *Under the γ-margin assumption, for a fixed $\gamma \in (1, \sqrt{3.4}]$, how many queries must be made in the SSAC framework for k-means to become tractable?*

Ashtiani *et al.* [6] addressed the above question and gave a query algorithm. Their algorithm, in fact, works for a more general setting where the clusters are not necessarily optimal. In the more general setting, there is a *target* clustering $\bar{X} = \bar{X}_1, \ldots, \bar{X}_k$ of the given dataset $X \subseteq \mathbb{R}^d$ such that these clusters satisfy the γ-margin property (i.e., for all $i, x \in \bar{X}_i$, and $y \notin \bar{X}_i, \gamma \cdot ||x - \mu(\bar{X}_i)|| < ||y - \mu(\bar{X}_i)||$), and the goal of the query algorithm is to output the clustering $\bar{X}$. Here is the main result of Ashtiani *et al.*

▶ **Theorem 4** (Theorems 7,8 in [6]). *Let $\delta \in (0, 1)$, $\gamma > 1$. Let $X \subseteq \mathbb{R}^d$ be any dataset with n points, k be a positive integer, and $X_1, \ldots, X_k$ be any target clustering of X that satisfies the γ-margin property. Then there is a query algorithm A that makes $O\left(k \log n + k^2 \frac{\log k + \log 1/\delta}{(\gamma-1)^4}\right)$ same-cluster queries and with probability at least $(1 - \delta)$, outputs the clustering $X_1, \ldots, X_k$. The running time of algorithm A is $O\left(kn \log n + k^2 \frac{\log k + \log 1/\delta}{(\gamma-1)^4}\right)$.*

The above result is a witness to the power of the SSAC framework. We extend this line of work by examining the power of same-cluster queries in the *approximation algorithms* domain. Our results do not assume any separation condition on the dataset (such as γ-margin as in [6]) and they hold for *any* dataset.

Since the k-means problem is NP-hard, an important line of research work has been to obtain approximation algorithms for the problem. There are many efficient approximation algorithms for the k-means problem, for example [25, 26]. Ahmadian *et al.* [2] gave a 6.357-approximation algorithm for the k-means problem. A simple approximation algorithm that gives an $O(\log k)$ approximation guarantee in expectation is the k-means++ seeding algorithm (also known as D^2-sampling algorithm) by Arthur and Vassilvitskii [5]. This algorithm is commonly used in solving the k-means problem in practice. As far as *approximation schemes* or in other words $(1 + \varepsilon)$-approximation algorithms (for arbitrary small $\varepsilon < 1$) are concerned, the following is known: It was shown by Awasthi *et al.* [9] that there is some fixed constant $0 < c < 1$ such that there cannot exist an efficient $(1 + c)$ factor approximation unless P = NP. This result was improved by Lee *et al.* [27] where it was shown that it is NP-hard to approximate the k-means problem within a factor of 1.0013. However, when either k or d is a fixed constant, then there are Polynomial Time Approximation Schemes (PTAS) for the k-means problem.[2] Addad *et al.* [15] and Friggstad *et al.* [19] gave PTASs for the k-means problem in constant dimension. For fixed constant k, various PTASs are known [26, 18, 23, 24]. Following is the main question that we explore in this work:

[1] If the optimal solution is not unique, the same-cluster query answers are assumed to be consistent with respect to any fixed optimal clustering.

[2] This basically means an algorithm that runs in time polynomial in the input parameters but may run in time exponential in $1/\varepsilon$.

For arbitrary small $\varepsilon > 0$, how many same-cluster queries must be made in an efficient $(1+\varepsilon)$-approximation algorithm for k-means in the SSAC framework? The running time should be polynomial in all input parameters such as n, k, d and also in $1/\varepsilon$.

Note that this is a natural extension of the main question explored by Ashtiani *et al.* [6]. Moreover, we have removed the separation assumption on the data. We provide an algorithm that makes $\text{poly}(k/\varepsilon)$ same-cluster queries and runs in time $O(nd \cdot \text{poly}(k/\varepsilon))$. More specifically, here is the formal statement of our main result:

▶ **Theorem 5** (Main result: query algorithm)**.** *Let $0 < \varepsilon \leq 1/2$. Let $X \subseteq \mathbb{R}^d$, and k be any positive integer. Then there is a query algorithm* A *that runs in time $O(ndk^9/\varepsilon^4)$, and with probability at least 0.99, outputs a center set C such that $\Phi(C, X) \leq (1+\varepsilon) \cdot \Delta_k(X)$. Moreover, the number of same-cluster queries used by* A *is $O(k^9/\varepsilon^4)$. Here $\Delta_k(X)$ denotes the optimal value of the k-means objective function.*

Note that unlike Theorem 4, our bound on the number of same-cluster queries is independent of the size of the dataset. We find this interesting and the next natural question we ask is whether this bound on the number of same-cluster queries is *tight* in some sense. In other words, does there exist a query algorithm in the SSAC setting that gives $(1+\varepsilon)$-approximation in time polynomial in n, k, d and makes significantly fewer queries than the one given in the theorem above? We answer this question in the negative by establishing a conditional lower bound on the number of same-cluster queries under the assumption that ETH (Exponential Time Hypothesis) [20, 21] holds. The formal statement of our result is given below.

▶ **Theorem 6** (Main result: query lower bound)**.** *If the Exponential Time Hypothesis (ETH) holds, then there exists a constant $c > 1$ such that any c-approximation query algorithm for the k-means problem that runs in time $\text{poly}(n, d, k)$ makes at least $\frac{k}{\text{poly}\log k}$ queries.*

1.1 Faulty query setting

The existence of a same-cluster oracle that answers such queries perfectly may be too strong an assumption. A more reasonable assumption is the existence of a *faulty* oracle that can answer incorrectly but only with bounded probability. Our query approximation algorithm can be extended to the setting where answers to the same-cluster queries are *faulty*. More specifically, we can get wrong answers to queries independently but with probability at most some constant $q < 1/2$. Also note that in our model the answer for a same-cluster query does not change with repetition. This means that one cannot ask the same query multiple times and amplify the probability of correctness. We obtain $(1+\varepsilon)$-approximation guarantee for the k-means clustering problem in this setting. The main result is given as follows.

▶ **Theorem 7.** *Let $0 < \varepsilon \leq 1/2$. Let $X \subseteq \mathbb{R}^d$, and k be any positive integer. Consider a faulty SSAC setting where the response to every same-cluster query is incorrect with probability at most some constant $q < 1/2$. In such a setting, there is a query algorithm A^E that with probability at least 0.99, outputs a center set C such that $\Phi(C, X) \leq (1+\varepsilon) \cdot \Delta_k(X)$. Moreover, the number of same-cluster queries used by A^E is $O(k^{15}/\varepsilon^8)$.*

The previous theorems summarise the main results of this work which basically explores the power of same-cluster queries in designing fast $(1+\varepsilon)$-approximation algorithms for the k-means problem. We will give the proofs of the above theorems in Sections 3, 4, and 5. There are some other simple and useful contexts, where the SSAC framework gives extremely nice results. One such context is the popular k-means++ seeding algorithm. This is an

extremely simple sampling based algorithm for the k-means problem that samples k centers in a sequence of k iterations. We show that within the SSAC framework, a small modification of this sampling algorithm converts it to one that gives constant factor approximation instead of $O(\log k)$-approximation [5] that is known. This is another witness to the power of same-cluster queries. We discuss this result in Section 2. Some of the basic techniques involved in proving our main results will be introduced while discussing this simpler context.

1.2 Other related work

Clustering problems have been studied in various semi-supervised settings. Basu *et al.* [12] explored *must-link* and *cannot-link* constraints in their semi-supervised clustering formulation. In their framework, must-link and cannot-link constraints were provided explicitly as part of the input along with the cost of violating these constraints. They gave an active learning formulation for clustering in which an oracle answers whether two query points belong to the same cluster or not, and gave a clustering algorithm using these queries. However, they work with a different objective function and there is no discussion on theoretical bounds on the number of queries. In contrast, in our work we consider the k-means objective function and provide bounds on approximation guarantee, required number of adaptive queries, and the running time. Balcan and Blum [10] proposed an interactive framework for clustering with *split/merge* queries. Given a clustering $C = \{C_1, \ldots\}$, a user provides feedback by specifying that some cluster C_l should be split, or clusters C_i and C_j should be merged. Awasthi *et al.* [7] studied a local interactive algorithm for clustering with split and merge feedbacks. Voevodski *et al.* [34] considered *one versus all* queries where query answer for a point $s \in X$ returns distances between s to all points in X. For a k-median instance satisfying (c, ε)-approximation stability property [11], the authors found a clustering close to true clustering using only $O(k)$ one versus all queries. Vikram and Dasgupta [33] designed an interactive bayesian hierarchical clustering algorithm. Given dataset X, the algorithm starts with a candidate hierarchy T, and an initially empty set C of constraints. The algorithm queries user with a subtree $T|_S$ of hierarchy T restricted to constant sized set $S \subset X$ of leaves. User either accepts $T|_S$ or provides a counterexample triplet $(\{a, b\}, c)$ which the algorithm adds to its set of constraints C, and updates T. They consider both random and adaptive ways to select S to query $T|_S$.

1.3 Our Techniques

We now give a brief outline of the new ideas needed for our results. Many algorithms for the k-means problem proceed by iteratively finding approximations to the optimal centers. One such popular algorithm is the k-means++ seeding algorithm [5]. In this algorithm, one builds a set of potential centers iteratively. We start with a set C initialized to the empty set. At each step, we sample a point with probability proportional to the square of the distance from C, and add it to C. Arthur and Vassilvitskii [5] showed that if we continue this process till $|C|$ reaches k, then the corresponding k-means solution has expected cost $O(\log k)$ times the optimal k-means cost. Aggarwal *et al.* [1] showed that if we continue this process till $|C|$ reaches βk, for some constant $\beta > 1$, then the corresponding k-means solution (where we actually open all the centers in C) has cost which is within constant factor of the optimal k-means cost with high probability. Ideally, one would like to stop when size of C reaches k and obtain a constant factor approximation guarantee. We know from previous works [5, 14, 13] that this is not possible in the classical (unsupervised) setting. In this work, we show that one can get such a result in the SSAC framework. A high-level

way of analysing the k-means++ seeding algorithm is as follows. We first observe that if we randomly sample a point from a cluster, then the expected cost of assigning all points of this cluster to the sampled point is within a constant factor of the cost of assigning all the points to the mean of this cluster. Therefore, it suffices to select a point chosen uniformly at random from each of the clusters. Suppose the set C contains such samples for the first i clusters (of an optimal solution). If the other clusters are far from these i clusters, then it is likely that the next point added to C belongs to a new cluster (and perhaps is close to a uniform sample). However to make this more probable, one needs to add several points to C. Further, the number of samples that needs to be added to C starts getting worse as the value of i increases. Therefore, the algorithm needs to build C till its size becomes $O(k \log k)$. In the SSAC framework, we can *tell* if the next point added in C belongs to a new cluster or not. Therefore, we can always ensure that $|C|$ does not exceed k. To make this idea work, we need to extend the induction argument of Arthur and Vassilvitskii [5] – details are given in Section 2.

We now explain the ideas for the PTAS for k-means. We consider the special case of $k = 2$. Let X_1 and X_2 denote the optimal clusters with X_1 being the larger cluster. Inaba *et al.* [22] showed that if we randomly sample about $O(1/\varepsilon)$ points from a cluster, and let μ' denote the mean of this subset of sampled points, then the cost of assigning all points in the cluster to μ' is within $(1 + \varepsilon)$ of the cost of assigning all these points to their actual mean (whp). Therefore, it is enough to get uniform samples of size about $O(1/\varepsilon)$ from each of the clusters. Jaiswal *et al.* [23] had the following approach for obtaining a $(1 + \varepsilon)$-approximation algorithm for k-means (with running time being $nd \cdot f(k, \varepsilon)$, where f is an exponential function of k/ε) – suppose we sample about $O(1/\varepsilon^2)$ points from the input, call this sample S. It is likely to contain at least $O(1/\varepsilon)$ from X_1, but we do not know which points in S are from X_1. Jaiswal *et al.* addressed this problem by cycling over all subsets of S. In the SSAC model, we can directly partition S into $S \cap X_1$ and $S \cap X_2$ using $|S|$ same-cluster queries. Having obtained such a sample S, we can get a close approximation to the mean of X_1. So assume for sake of simplicity that we know μ_1, the mean of X_1. Now we are faced with the problem of obtaining a uniform sample from X_2. The next idea of Jaiswal *et al.* is to sample points with probability proportional to square of distance from μ_1. This is known as D^2-sampling. Suppose we again sample about $O(1/\varepsilon^2)$ such points, call this sample S'. Assuming that the two clusters are far enough (otherwise the problem only gets easier), they show that S' will contain about $O(1/\varepsilon^2)$ points from X_2 (with good probability). Again, in the SSAC model, we can find this subset by $|S'|$ queries – call this set S''. However, the problem is that S'' may not represent a uniform sample from X_2. For any point $e \in X_2$, let p_e denote the conditional probability of sampling e given that a point from X_2 is sampled when sampled using D^2-sampling. They showed p_e is at least $\frac{\varepsilon}{m}$, where m denotes the size of X_2. In order for the sampling lemma of Inaba *et al.* [22] to work, we cannot work with approximately uniform sampling. The final trick of Jaiswal *et al.* was to show that one can in fact get a uniform sample of size about $O(\varepsilon|S''|) = O(1/\varepsilon)$ from S''. The idea is as follows – for every element $e \in S''$, we *retain* it with probability $\frac{\varepsilon}{p_e m}$ (which is at most 1), otherwise we remove it from S''. It is not difficult to see that this gives a uniform sample from X_2. The issue is that we do not know m. Jaiswal *et al.* again cycle over all subsets of S' – we know that there is a (large enough) subset of S' which will behave like a uniform sample from X_2. In the SSAC framework, we first identify the subset of S' which belongs to X_2, call this S'' (as above). Now we prune some points from S'' such that the remaining points behave like a uniform sample. This step is non-trivial because as indicated above, we do not know the value m. Instead, we first show that p_e lies between $\frac{\varepsilon}{m}$ and $\frac{2}{m}$ for most of the

Table 1 k-means++ seeding algorithm (left) and its adaptation in the SSAC setting (right).

k-means++(X,k)	Query-k-means++(X,k)
- Randomly sample a point $x \in X$ - $C \leftarrow \{x\}$ - for $i = 2$ to k - Sample $x \in X$ using distribution D, where $D(x) = \frac{\Phi(C,\{x\})}{\Phi(C,X)}$ - $C \leftarrow C \cup \{x\}$ - return(C)	- Randomly sample a point $x \in X$ - $C \leftarrow \{x\}$ - for $i = 2$ to k - for $j = 1$ to $\lceil \log k \rceil$ - Sample $x \in X$ using distribution D, where $D(x) = \frac{\Phi(C,\{x\})}{\Phi(C,X)}$ - if (NewCluster(C,x)) {$C \leftarrow C \cup \{x\}$; break} - return$(C)$
	NewCluster(C,x) - If ($\exists c \in C$ s.t. SameCluster(c,x)) return(false) - else return(true)

points of X_2. Therefore, S'' is likely to contain such a *nice* point, call it v. Now, for every point $e \in S''$, we retain it with probability $\frac{\varepsilon p_e}{2p_v}$ (which we know is at most 1). This gives a uniform sample of sufficiently large size from X_2. For k larger than 2, we generalize the above ideas using a non-trivial induction argument.

2 k-means++ within SSAC framework

The k-means++ seeding algorithm, also known as the D^2-sampling algorithm, is a simple sampling procedure that samples k centers in k iterations. The description of this algorithm is given below.

The algorithm picks the first center randomly from the set X of points and after having picked the first $(i-1)$ centers denoted by C_{i-1}, it picks a point $x \in X$ to be the i^{th} center with probability proportional to $\min_{c \in C_{i-1}} ||x-c||^2$. The running time of k-means++ seeding algorithm is clearly $O(nkd)$. Arthur and Vassilvitskii [5] showed that this simple sampling procedure gives an $O(\log k)$ approximation in expectation for any dataset. Within the SSAC framework where the algorithm is allowed to make same-cluster queries, we can make a tiny addition to the k-means++ seeding algorithm to obtain a query algorithm that gives constant factor approximation guarantee and makes only $O(k^2 \log k)$ same-cluster queries. The description of the query algorithm is given in Table 1 (see right). In iteration $i > 1$, instead of simply accepting the sampled point x as the i^{th} center (as done in k-means++ seeding algorithm), the sampled point x is accepted only if it belongs to a cluster other than those to which centers in C_{i-1} belong (if this does not happen, the sampling is repeated for at most $\lceil \log k \rceil$ times). Here is the main result that we show for the query-k-means++ algorithm.

▶ **Theorem 8.** *Let $X \subseteq \mathbb{R}^d$ be any dataset containing n points and $k > 1$ be a positive integer. Let C denote the output of the algorithm* Query-k-means++(X,k). *Then*

$$\mathbf{E}[\Phi(C,X)] \leq 24 \cdot \Delta_k(X),$$

where $\Delta_k(X)$ denotes the optimal k-means cost on dataset X. Furthermore, the algorithm makes $O(k^2 \log k)$ same-cluster queries and runs in time $O(nkd + k \log k \log n + k^2 \log k)$.

The bound on the number of same-cluster queries is trivial from the algorithm description. For the running time, it takes $O(nd)$ time to update the distribution D which is updated k times. This accounts for the $O(nkd)$ term in the running time. Sampling an element from a distribution D takes $O(\log n)$ time (if we maintain the cumulative distribution etc.) and at most $O(k \log k)$ points are sampled. Moreover, determining whether a sampled point belongs to an uncovered cluster takes $O(k)$ time. So, the overall running time of the algorithm is $O(nkd + k \log k \log n + k^2 \log k)$. The proof of the above theorem may be found in the full version of the paper available at [3].

3 Query Approximation Algorithm (proof of Theorem 5)

As mentioned in the introduction, our query algorithm is based on the D^2-sampling based algorithm of Jaiswal *et al.* [23, 24]. The algorithm in these works give a $(1+\varepsilon)$-factor approximation for arbitrary small $\varepsilon > 0$. The running time of these algorithms are of the form $nd \cdot f(k, \varepsilon)$, where f is an exponential function of k/ε. We now show that it is possible to get a running time which is polynomial in $n, k, d, 1/\varepsilon$ in the SSAC model. The main ingredient in the design and analysis of the sampling algorithm is the following lemma by Inaba *et al.* [22].

▶ **Lemma 9** ([22]). *Let S be a set of points obtained by independently sampling M points uniformly at random with replacement from a point set $X \subset \mathbb{R}^d$. Then for any $\delta > 0$,*

$$\mathbf{Pr}\left[\Phi(\{\mu(S)\}, X) \leq \left(1 + \frac{1}{\delta M}\right) \cdot \Delta_1(X)\right] \geq (1 - \delta).$$

Here $\mu(S)$ denotes the geometric centroid of the set S. That is $\mu(S) = \frac{\sum_{s \in S} s}{|S|}$

Our algorithm `Query-`k`-means` is described in Table 2. It maintains a set C of potential centers of the clusters. In each iteration of step **(3)**, it adds one more candidate center to the set C (whp), and so, the algorithm stops when $|C|$ reaches k. For sake of explanation, assume that optimal clusters are $X_1, X_2, \ldots, X_k$ with means $\mu_1, \ldots, \mu_k$ respectively. Consider the i^{th} iteration of step **(3)**. At this time $|C| = i - 1$, and it has good approximations to means of $i-1$ clusters among $X_1, \ldots . X_k$. Let us call these clusters *covered*. In Step **(3.1)**, it samples N points, each with probability proportional to square of distance from C (D^2-sampling). Now, it partitions this set, S, into $S \cap X_1, \ldots, S \cap X_k$ in the procedure `UncoveredCluster`, and then picks the partition with the largest size such that the corresponding optimal cluster X_j is not one of the $(i-1)$ covered clusters. Now, we would like to get a uniform sample from X_j – recall that $S \cap X_j$ does not represent a uniform sample. However, as mentioned in the introduction, we need to find an element s of X_j for which the probability of being in sampled is small enough. Therefore, we pick the element in $S \cap X_j$ for which this probability is smallest (and we will show that it has the desired properties). The procedure `UncoveredCluster` returns this element s. Finally, we choose a subset T of $S \cap X_j$ in the procedure `UniformSample`. This procedure is designed such that each element of X_j has the same probability of being in T. In step **(3.4)**, we check whether the multi-set T is of a desired minimum size. We will argue that the probability of T not containing sufficient number of points is very small. If we have T of the desired size, we take its mean and add it to C in Step **(3.6)**.

We now formally prove the approximation guarantee of the `Query-`k`-means` algorithm.

Table 2 Approximation algorithm for k-means(top-left frame). Note that $\mu(T)$ denotes the centroid of T and D^2-sampling w.r.t. empty center set C means just uniform sampling. The algorithm `UniformSample`(X, C, s) (bottom-left) returns a uniform sample of size $\Omega(1/\varepsilon)$ (w.h.p.) from the optimal cluster to which point s belongs.

Constants: $N = \frac{(2^{12})k^3}{\varepsilon^2}$, $M = \frac{64k}{\varepsilon}$, $L = \frac{(2^{23})k^2}{\varepsilon^4}$

`Query-`k`-means`(X, k, ε)

(1) $R \leftarrow \emptyset$

(2) $C \leftarrow \emptyset$

(3) for $i = 1$ to k

(3.1) D^2-sample a multi-set S of N points from X with respect to center set C

(3.2) $s \leftarrow$ `UncoveredCluster`(C, S, R)

(3.3) $T \leftarrow$ `UniformSample`(X, C, s)

(3.4) If $(|T| < M)$ continue

(3.5) $R \leftarrow R \cup \{s\}$

(3.6) $C \leftarrow C \cup \mu(T)$

(4) return(C)

`UniformSample`(X, C, s)

- $T \leftarrow \emptyset$
- For $i = 1$ to L:
  - D^2-sample a point x from X with respect to center set C
  - If (`SameCluster`(s, x))
    - With probability $\left(\frac{\varepsilon}{128} \cdot \frac{\Phi(C,\{s\})}{\Phi(C,\{x\})}\right)$ add x in multi-set T
- return(T)

`UncoveredCluster`(C, S, R)

- For all $i \in \{1, ..., k\}$: $S_i \leftarrow \emptyset$
- $i \leftarrow 1$
- For all $y \in R$: $\{S_i \leftarrow y;\ i{+}{+}\}$
- For all $s \in S$:
  - If $(\exists j, y$ s.t. $y \in S_j$ & `SameCluster`$(s, y))$
    - $S_j \leftarrow S_j \cup \{s\}$
  - else
    - Let i be any index s.t. S_i is empty
    - $S_i \leftarrow \{s\}$.
- Let S_i be the largest set such that $i > |R|$.
- Let $s \in S_i$ be the element with smallest
  - value of $\Phi(C, \{s\})$ in S_i.
- return(s)

▶ **Theorem 10.** *Let $0 < \varepsilon \leq 1/2$. Let $X \subseteq \mathbb{R}^d$, and k be any positive integer. There exists an algorithm that runs in time $O(ndk^9/\varepsilon^4)$, and with probability at least $\frac{1}{4}$, outputs a center set C such that $\Phi(C, X) \leq (1 + \varepsilon) \cdot \Delta_k(X)$. Moreover, the algorithm makes $O(k^9/\varepsilon^4)$ same-cluster queries.*

Note that the success probability of the algorithm may be boosted by repeating it a constant number of times. This will also prove our main theorem (that is, Theorem 5). We will assume that the dataset X satisfies (k, ε)-irreducibility property defined next. We will later drop this assumption using a simple argument and show that the result holds for *all* datasets. This property was also used in some earlier works [26, 23].

▶ **Definition 11** (Irreducibility)**.** Let k be a positive integer and $0 < \varepsilon \leq 1$. A given dataset $X \subseteq \mathbb{R}^d$ is said to be (k, ε)-irreducible iff

$$\Delta_{k-1}(X) \geq (1 + \varepsilon) \cdot \Delta_k(X).$$

Qualitatively, what the irreducibility assumption implies is that the optimal solution for the $(k - 1)$-means problem does not give a $(1 + \varepsilon)$-approximation to the k-means problem.

The following useful lemmas are well known facts.

▶ **Lemma 12.** *For any dataset $X \subseteq \mathbb{R}^d$ and a point $c \in \mathbb{R}^d$, we have:*

$$\Phi(\{c\}, X) = \Phi(\mu(X), X) + |X| \cdot ||c - \mu(X)||^2.$$

▶ **Lemma 13** (Approximate Triangle Inequality)**.** *For any three points $p, q, r \in \mathbb{R}^d$, we have*

$$||p - q||^2 \leq 2(||p - r||^2 + ||r - q||^2)$$

Let $X_1, ..., X_k$ be optimal clusters of the dataset X for the k-means objective. Let $\mu_1, ..., \mu_k$ denote the corresponding optimal k centers. That is, $\forall i, \mu_i = \mu(X_i)$. For all i, let $m_i = |X_i|$. Also, for all i, let $r_i = \frac{\sum_{x \in X_i} ||x - \mu_i||^2}{m_i}$. The following useful lemma holds due to irreducibility. The lemma is the same as Lemma 4 from [23].

▶ **Lemma 14.** *For all $1 \leq i < j \leq k$, $||\mu_i - \mu_j||^2 \geq \varepsilon \cdot (r_i + r_j)$.*

Consider the algorithm `Query-k-means` in Figure 2. Let $C_i = \{c_1, ..., c_i\}$ denote the set of centers at the end of the i^{th} iteration of the for loop. That is, C_i is the same as variable C at the end of iteration i. We will prove Theorem 10 by inductively arguing that for every i, there are i distinct clusters for which centers in C_i are good in some sense. Consider the following invariant:

P(i): There exists a set of i distinct clusters $X_{j_1}, X_{j_2}, ..., X_{j_i}$ such that

$$\forall r \in \{1, ..., i\}, \Phi(\{c_r\}, X_{j_r}) \leq \left(1 + \frac{\varepsilon}{16}\right) \cdot \Delta_1(X_{j_r}).$$

Note that a trivial consequence of $P(i)$ is $\Phi(C_i, X_{j_1} \cup ... \cup X_{j_i}) \leq (1 + \frac{\varepsilon}{16}) \cdot \sum_{r=1}^{i} \Delta_1(X_{j_r})$. We will show that for all i, $P(i)$ holds with probability at least $(1 - 1/k)^i$. Note that the theorem follows if $P(k)$ holds with probability at least $(1 - 1/k)^k$. We will proceed using induction.

The base case $P(0)$ holds since C_0 is the empty set. For the inductive step, assuming that $P(i)$ holds with probability at least $(1 - 1/k)^i$ for some arbitrary $i \geq 0$, we will show that $P(i + 1)$ holds with probability at least $(1 - 1/k)^{i+1}$. We condition on the event $P(i)$ (that is true with probability at least $(1 - 1/k)^i$). Let C_i and $X_{j_1}, ..., X_{j_i}$ be as guaranteed by the invariant $P(i)$. For ease of notation and without loss of generality, let us assume that the index j_r is r. So, C_i gives a good approximation w.r.t. points in the set $X_1 \cup X_2 \cup .... \cup X_i$ and these clusters may be thought of as "covered" clusters (in the approximation sense). Suppose we D^2-sample a point p w.r.t. center set C_i. The probability that p belongs to some "uncovered cluster" X_r where $r \in [i + 1, k]$ is given as $\frac{\Phi(C_i, X_r)}{\Phi(C_i, X)}$. If this quantity is small, then the points sampled using D^2 sampling in subsequent iterations may not be good representatives for the uncovered clusters. This may cause the analysis to break down. However, we argue that since our data is (k, ε)-irreducible, this does not occur. The following lemma is the same as Lemma 5 from [23].

▶ **Lemma 15.** $\frac{\Phi(C_i, X_{i+1} \cup ... \cup X_k)}{\Phi(C_i, X)} \geq \frac{\varepsilon}{4}$.

The following simple corollary of the above lemma will be used in the analysis later.

▶ **Corollary 16.** *There exists an index $j \in \{i + 1, ..., k\}$ such that $\frac{\Phi(C_i, X_j)}{\Phi(C_i, X)} \geq \frac{\varepsilon}{4k}$.*

The above corollary says that there is an uncovered cluster which will have a non-negligible representation in the set S that is sampled in iteration $(i+1)$ of the algorithm `Query-`k`-means`. The next lemma shows that conditioned on sampling from an uncovered cluster $l \in \{i+1, ..., k\}$, the probability of sampling a point x from X_l is at least $\frac{\varepsilon}{64}$ times its sampling probability if it were sampled uniformly from X_l (i.e., with probability at least $\frac{\varepsilon}{64} \cdot \frac{1}{m_l}$). [3]

▶ **Lemma 17.** *For any $l \in \{i+1, ..., k\}$ and $x \in X_l$, $\frac{\Phi(C_i, \{x\})}{\Phi(C_i, X_l)} \geq \frac{\varepsilon}{64} \cdot \frac{1}{m_l}$.*

Proof. Let $t \in \{1, ..., i\}$ be the index such that x is closest to c_t among all centers in C_i. We have:

$$\begin{aligned}
\Phi(C_i, X_l) &= m_l \cdot r_l + m_l \cdot ||\mu_l - c_t||^2 \quad \text{(using Lemma 12)} \\
&\leq m_l \cdot r_l + 2m_l \cdot (||\mu_l - \mu_t||^2 + ||\mu_t - c_t||^2) \quad \text{(using Lemma 13)} \\
&\leq m_l \cdot r_l + 2m_l \cdot (||\mu_l - \mu_t||^2 + \frac{\varepsilon}{16} \cdot r_t) \quad \text{(using invariant and Lemma 12)}
\end{aligned}$$

Also, we have:

$$\begin{aligned}
\Phi(C_i, \{x\}) = ||x - c_t||^2 &\geq \frac{||x - \mu_t||^2}{2} - ||\mu_t - c_t||^2 \quad \text{(using Lemma 13)} \\
&\geq \frac{||\mu_l - \mu_t||^2}{8} - ||\mu_t - c_t||^2 \quad \text{(since } ||x - \mu_t|| \geq ||\mu_l - \mu_t||/2\text{)} \\
&\geq \frac{||\mu_l - \mu_t||^2}{8} - \frac{\varepsilon}{16} \cdot r_t \quad \text{(using invariant and Lemma 12)} \\
&\geq \frac{||\mu_l - \mu_t||^2}{16} \quad \text{(using Lemma 14)}
\end{aligned}$$

Combining the inequalities obtained above, we get the following:

$$\begin{aligned}
\frac{\Phi(C_i, \{x\})}{\Phi(C_i, X_l)} &\geq \frac{||\mu_l - \mu_t||^2}{16 \cdot m_l \cdot \left(r_l + 2||\mu_l - \mu_t||^2 + \frac{\varepsilon}{8} \cdot r_t\right)} \\
&\geq \frac{1}{16 \cdot m_l} \cdot \frac{1}{(1/\varepsilon) + 2 + (1/8)} \geq \frac{\varepsilon}{64} \cdot \frac{1}{m_l}
\end{aligned}$$

This completes the proof of the lemma. ◀

With the above lemmas in place, let us now get back to the inductive step of the proof. Let $J \subseteq \{i+1, ..., k\}$ denote the subset of indices (from the uncovered cluster indices) such that $\forall j \in J, \frac{\Phi(C_i, X_j)}{\Phi(C_i, X)} \geq \frac{\varepsilon}{8k}$. For any index $j \in J$, let $Y_j \subseteq X_j$ denote the subset of points in X_j such that $\forall y \in Y_j, \frac{\Phi(C_i, \{y\})}{\Phi(C_i, X_j)} \leq \frac{2}{m_j}$. That is, Y_j consists of all the points such that the conditional probability of sampling any point y in Y_j, given that a point is sampled from X_j, is upper bounded by $2/m_j$. Note that from Lemma 17, the conditional probability of sampling a point x from X_j, given that a point is sampled from X_j, is lower bounded by $\frac{\varepsilon}{64} \cdot \frac{1}{m_j}$. This gives the following simple and useful lemma.

▶ **Lemma 18.** *For all $j \in \{i+1, ..., k\}$ the following two inequalities hold:*
1. *$\frac{\Phi(C_i, Y_j)}{\Phi(C_i, X)} \geq \frac{\varepsilon}{128} \cdot \frac{\Phi(C_i, X_j)}{\Phi(C_i, X)}$, and*
2. *For any $y \in Y_j$ and any $x \in X_j$, $\frac{\varepsilon}{128} \cdot \Phi(C_i, \{y\}) \leq \Phi(C_i, \{x\})$.*

[3] This is Lemma 6 from [23]. We give the proof for self-containment.

Proof. Inequality (1) follows from the fact that $|Y_j| \geq m_j/2$, and $\frac{\Phi(C_i,\{y\})}{\Phi(C_i,X_j)} \geq \frac{\varepsilon}{64} \cdot \frac{1}{m_j}$ for all $y \in X_j$. Inequality (2) follows from the fact that for all $x \in X_j$, $\frac{\Phi(C_i,\{x\})}{\Phi(C_i,X_j)} \geq \frac{\varepsilon}{64} \cdot \frac{1}{m_j}$ and for all $y \in Y_j$, $\frac{\Phi(C_i,\{y\})}{\Phi(C_i,X_j)} \leq \frac{2}{m_j}$. ◀

Let us see the outline of our plan before continuing with our formal analysis. What we hope to get in line (3.2) of the algorithm is a point s that belongs to one of the uncovered clusters with index in the set J. That is, s belongs to an uncovered cluster that is likely to have a good representation in the D^2-sampled set S obtained in line (3.1). In addition to s belonging to X_j for some $j \in J$, we would like s to belong to Y_j. This is crucial for the uniform sampling in line (3.3) to succeed. We will now show that the probability of s returned in line (3.2) satisfies the above conditions is large.

▶ **Lemma 19.** *Let S denote the D^2-sample obtained w.r.t. center set C_i in line (3.1) of the algorithm.*

$$\mathbf{Pr}[\exists j \in J \text{ such that } S \text{ does not contain any point from } Y_j] \leq \frac{1}{4k}.$$

Proof. We will first get bound on the probability for a fixed $j \in J$ and then use the union bound. From property (1) of Lemma 18, we have that for any $j \in J$, $\frac{\Phi(C_i,Y_j)}{\Phi(C_i,X)} \geq \frac{\varepsilon}{128} \cdot \frac{\varepsilon}{8k} = \frac{\varepsilon^2}{(2^{10})k}$. Since the number of sampled points is $N = \frac{(2^{12})k^3}{\varepsilon^2}$, we get that the probability that S has no points from Y_j is at most $\frac{1}{4k^2}$. Finally, using the union bound, we get the statement of the lemma. ◀

▶ **Lemma 20.** *Let S denote the D^2-sample obtained w.r.t. center set C_i in line (3.1) of the algorithm and let S_j denote the representatives of X_j in S. Let $max = \arg\max_{j \in \{i+1,\ldots,k\}} |S_j|$. Then $\mathbf{Pr}[max \notin J] \leq \frac{1}{4k}$.*

Proof. From Corollary 16, we know that there is an index $j \in \{i+1,\ldots,k\}$ such that $\frac{\Phi(C_i,X_j)}{\Phi(C_i,X)} \geq \frac{\varepsilon}{4k}$. Let $\alpha = N \cdot \frac{\varepsilon}{4k}$. The expected number of representatives from X_j in S is at least α. So, from Chernoff bounds, we have:

$$\mathbf{Pr}[|S_j| \leq 3\alpha/4] \leq e^{-\alpha/32}$$

On the other hand, for any $r \in \{i+1,\ldots,k\} \setminus J$, the expected number of points in S from X_r is at most $\frac{\varepsilon}{8k} \cdot N = \alpha/2$. So, from Chernoff bounds, we have:

$$\mathbf{Pr}[|S_r| > 3\alpha/4] \leq e^{-\alpha/24}$$

So, the probability that there exists such an r is at most $k \cdot e^{-\alpha/24}$ by union bound. Finally, the probability that $max \notin J$ is at most $\mathbf{Pr}[|S_j| \leq 3\alpha/4] + \mathbf{Pr}[\exists r \in \{i+1,\ldots,k\} \setminus J | |S_r| > 3\alpha/4]$ which is at most $\frac{1}{4k}$ due to our choice of $N = \frac{(2^{12})k^3}{\varepsilon^2}$. ◀

From the previous two lemmas, we get that with probability at least $(1 - \frac{1}{2k})$, the s returned in line (3.2) belongs to Y_j for some $j \in J$. Finally, we will need the following claim to argue that the set T returned in line (3.3) is a uniform sample from one of the sets X_j for $j \in \{i+1,\ldots,k\}$.

▶ **Lemma 21.** *Let S denote the D^2-sample obtained w.r.t. center set C_i in line (3.1) and s be the point returned in line (3.2) of the algorithm. Let j denote the index of the cluster to which s belongs. If $j \in J$ and $s \in Y_j$, then with probability at least $(1 - \frac{1}{4k})$, T returned in line (3.3) is a uniform sample from X_j with size at least $\frac{64k}{\varepsilon}$.*

Proof. Consider the call to sub-routine `UniformSample`(X, C_i, s) with s as given in the statement of the lemma. If j is the index of the cluster to which s belongs, then $j \in J$ and $s \in Y_j$. Let us define L random variables $Z_1, ..., Z_L$ one for every iteration of the sub-routine. These random variables are defined as follows: for any $r \in [1, L]$, if the sampled point x belongs to the same cluster as s and it is picked to be included in multi-set S, then $Z_r = x$, otherwise $Z_r = \perp$. We first note that for any r and any $x, y \in X_j$, $\mathbf{Pr}[Z_r = x] = \mathbf{Pr}[Z_r = y]$. This is because for any $x \in X_j$, we have $\mathbf{Pr}[Z_r = x] = \frac{\Phi(C_i,\{x\})}{\Phi(C_i,X)} \cdot \frac{\frac{\varepsilon}{128}\cdot\Phi(C_i,\{s\})}{\Phi(C_i,\{x\})} = \frac{\varepsilon}{128} \cdot \frac{\Phi(C_i,\{s\})}{\Phi(C_i,X)}$. It is important to note that $\frac{\frac{\varepsilon}{128}\cdot\Phi(C_i,\{s\})}{\Phi(C_i,\{x\})} \leq 1$ from property (2) of Lemma 18 and hence valid in the probability calculations above.

Let us now obtain a bound on the size of T. Let $T_r = I(Z_r)$ be the indicator variable that is 1 if $Z_r \neq \perp$ and 0 otherwise. Using the fact that $j \in J$, we get that for any r:

$$\mathbf{E}[T_r] = \mathbf{Pr}[T_r = 1] = \frac{\varepsilon}{128} \cdot \frac{\sum_{x \in X_j} \Phi(C_i, \{s\})}{\Phi(C_i, X)} \geq \frac{\varepsilon}{128} \cdot \frac{\varepsilon}{8k} \cdot \frac{\varepsilon}{64} = \frac{\varepsilon^3}{(2^{16})k}.$$

Given that $L = \frac{2^{23}k^2}{\varepsilon^4}$, applying Chernoff bounds, we get the following:

$$\mathbf{Pr}\left[|T| \geq \frac{64k}{\varepsilon}\right] = 1 - \mathbf{Pr}\left[|T| \leq \frac{64k}{\varepsilon}\right] \geq \left(1 - \frac{1}{4k}\right)$$

This completes the proof of the lemma. ◀

Since a suitable s (as required by the above lemma) is obtained in line (3.2) with probability at least $(1 - \frac{1}{2k})$, the probability that T obtained in line (3.3) is a uniform sample from some uncovered cluster X_j is at least $(1 - \frac{1}{2k}) \cdot (1 - \frac{1}{4k})$. Finally, the probability that the centroid $\mu(T)$ of the multi-set T that is obtained is a good center for X_j is at least $(1 - \frac{1}{4k})$ using Inaba's lemma. Combining everything, we get that with probability at least $(1 - \frac{1}{k})$ an uncovered cluster will be covered in the i^{th} iteration. This completes the inductive step and hence the approximation guarantee of $(1 + \varepsilon)$ holds for any dataset that satisfies the (k, ε)-irreducibility assumption. For the number of queries and running time, note that every time sub-routine `UncoveredCluster` is called, it uses at most kN same cluster queries. For the subroutine `UniformSample`, the number of same-cluster queries made is L. So, the total number of queries is $O(k(kN + L)) = O(k^5/\varepsilon^4)$. More specifically, we have proved the following theorem.

▶ **Theorem 22.** *Let $0 < \varepsilon \leq 1/2$, k be any positive integer, and $X \subseteq \mathbb{R}^d$ such that X is (k, ε)-irreducible. Then* `Query-k-means`(X, k, ε) *runs in time $O(ndk^5/\varepsilon^4)$ and with probability at least $(1/4)$ outputs a center set C such that $\Phi(C, X) \leq (1+\varepsilon) \cdot \Delta_k(X)$. Moreover, the number of same-cluster queries used by* `Query-k-means`(X, k, ε) *is $O(k^5/\varepsilon^4)$.*

To complete the proof of Theorem 10, we must remove the irreducibility assumption in the above theorem. We do this by considering the following two cases:

1. Dataset X is $(k, \frac{\varepsilon}{(4+\varepsilon/2)k})$-irreducible.
2. Dataset X is not $(k, \frac{\varepsilon}{(4+\varepsilon/2)k})$-irreducible.

In the former case, we can apply Theorem 22 to obtain Theorem 10. Now, consider the latter case. Let $1 < i \leq k$ denote the largest index such that X is $(i, \frac{\varepsilon}{(1+\varepsilon/2)k})$-irreducible, otherwise $i = 1$. Then we have:

$$\Delta_i(X) \leq \left(1 + \frac{\varepsilon}{(4 + \varepsilon/2)k}\right)^{k-i} \cdot \Delta_k(X) \leq \left(1 + \frac{\varepsilon}{4}\right) \cdot \Delta_k(X).$$

This means that a $(1+\varepsilon/4)$-approximation for the i-means problem on the dataset X gives the desired approximation for the k-means problem. Note that our approximation analysis works for the i-means problem with respect to the algorithm being run only for i steps in line (3) (instead of k). That is, the centers sampled in the first i iterations of the algorithm give a $(1+\varepsilon/16)$-approximation for the i-means problem for any fixed i. This simple observation is sufficient for Theorem 10.

Note since Theorem 22 is used with value of the error parameter as $O(\varepsilon/k)$, the bounds on the query and running time get multiplied by a factor of k^4.

4 Query Lower Bound (proof of Theorem 6)

In this section, we will obtain a conditional lower bound on the number of same-cluster queries assuming the Exponential Time Hypothesis (ETH). This hypothesis has been used to obtain lower bounds in various different contexts (see [29] for reference). We start by stating the Exponential Time Hypothesis (ETH).

▶ **Hypothesis 23** (Exponential Time Hypothesis (ETH)[20, 21])**.** *There does not exist an algorithm that can decide whether any 3-SAT formula with m clauses is satisfiable with running time $2^{o(m)}$.*

Since we would like to obtain lower bounds in the approximation domain, we will need a *gap* version of the above ETH hypothesis. The following version of the PCP theorem will be very useful in obtaining a gap version of ETH.

▶ **Theorem 24** (Dinur's PCP Theorem [17])**.** *For some constants $\varepsilon, d > 0$, there exists a polynomial time reduction that takes a 3-SAT formula ψ with m clauses and produces another 3-SAT formula ϕ with $m' = O(m \text{ polylog } m)$ clauses such that:*

- *If ψ is satisfiable, then ϕ is satisfiable,*
- *if ψ is unsatisfiable, then $val(\phi) \leq 1-\varepsilon$, and*
- *each variable of ϕ appears in at most d clauses.*

Here $val(\phi)$ denotes the maximum fraction of clauses of ϕ that are satisfiable by any assignment.

The following new hypothesis follows from ETH and will be useful in our analysis.

▶ **Hypothesis 25.** *There exists constants $\varepsilon, d > 0$ such that the following holds: There does not exist an algorithm that, given a 3-SAT formula ψ with m clauses with each variable appearing in at most d clauses, distinguishes whether ψ is satisfiable or $val(\psi) \leq (1-\varepsilon)$, runs in time $2^{\Omega(\frac{m}{poly \log m})}$.*

The following simple lemma follows from Dinur's PCP theorem given above.

▶ **Lemma 26.** *If Hypothesis 23 holds, then so does Hypothesis 25.*

We now see a reduction from the gap version of 3-SAT to the gap version of the Vertex Cover problem that will be used to argue the hardness of the k-means problem. The next result is a standard reduction and can be found in a survey by Luca Trevisan [31].

▶ **Lemma 27.** *Let $\varepsilon, d > 0$ be some constants. There is a polynomial time computable function mapping 3-SAT instances ψ with m variables and where each variable appears in at most d clauses, into graphs G_ψ with $3m$ vertices and maximum degree $O(d)$ such that if ψ is satisfiable, then G_ψ has a vertex cover of size at most $2m$ and if $val(\psi) \leq (1-\varepsilon)$, then every vertex cover of G_ψ has size at least $2m(1+\varepsilon/2)$.*

We formulate the following new hypothesis that holds given that hypothesis 25 holds. Eventually, we will chain all these hypothesis together.

▶ **Hypothesis 28.** *There exists constants $\varepsilon, d > 0$ such that the following holds: There does not exist an algorithm that, given a graph G with n vertices and maximum degree d, distinguishes between the case when G has a vertex cover of size at most $2n/3$ and the case when G has a vertex cover of size at least $\frac{2n}{3} \cdot (1+\varepsilon)$, runs in time $2^{\Omega(\frac{n}{poly \log n})}$.*

The following lemma is a simple implication of Lemma 27

▶ **Lemma 29.** *If Hypothesis 25 holds, then so does Hypothesis 28.*

We are getting closer to the k-means problem that has a reduction from the vertex cover problem on triangle-free graphs [9]. So, we will need reductions from vertex cover problem to vertex cover problem on triangle-free graphs and then to the k-means problem. These two reductions are given by Awasthi *et al.* [9].

▶ **Lemma 30** (Follows from Theorem 21 [9])**.** *Let $\varepsilon, d > 0$ be some constants. There is a polynomial-time computable function mapping any graph $G = (V, E)$ with maximum degree d to a triangle-free graph $\hat{G} = (\hat{V}, \hat{E})$ such that the following holds:*

- *$|\hat{V}| = poly(d, 1/\varepsilon) \cdot |V|$ and maximum degree of vertices in $\hat{G}$ is $O(d^3/\varepsilon^2)$, and*
- *$\left(1 - \frac{|VC(G)|}{|V|}\right) \leq \left(1 - \frac{|VC(\hat{G})|}{|\hat{V}|}\right) \leq (1+\varepsilon) \cdot \left(1 - \frac{|VC(G)|}{|V|}\right)$.*

Here $VC(G)$ denote the size of the minimum vertex cover of G.

We can formulate the following hypothesis that will follow from Hypothesis 28 using the above lemma.

▶ **Hypothesis 31.** *There exists constants $\varepsilon, d > 0$ such that the following holds: There does not exist an algorithm that, given a triangle-free graph G with n vertices and maximum degree d, distinguishes between the case when G has a vertex cover of size at most $\frac{2n}{3}$ and the case when G has a vertex cover of size at least $\frac{2n}{3} \cdot (1+\varepsilon)$, runs in time $2^{\Omega(\frac{n}{poly \log n})}$.*

The next claim is a simple application of Lemma 30.

▶ **Lemma 32.** *If Hypothesis 28 holds, then so does Hypothesis 31.*

Finally, we use the reduction from the vertex cover problem in triangle-free graphs to the k-means problem to obtain the hardness result for the k-means problem. We will use the following reduction from Awasthi *et al.* [9].

▶ **Lemma 33** (Theorem 3 [9])**.** *There is an efficient reduction from instances of Vertex Cover (in triangle free graphs) to those of k-means that satisfies the following properties:*

- *if the Vertex Cover instance has value k, then the k-means instance has cost at most $(m-k)$*
- *if the Vertex Cover instance has value at least $k(1+\varepsilon)$, then the optimal k-means cost is at least $m - (1 - \Omega(\varepsilon))k$. Here ε is some fixed constant > 0.*

Here m denotes the number of edges in the vertex cover instance.

The next hypothesis follows from Hypothesis 31 due to the above lemma.

▶ **Hypothesis 34.** *There exists constant $c > 1$ such that the following holds: There does not exist an algorithm that gives an approximation guarantee of c for the k-means problem that runs in time $poly(n, d) \cdot 2^{\Omega(\frac{k}{poly \log k})}$.*

▶ **Lemma 35.** *If Hypothesis 31 holds, then so does Hypothesis 34.*

Now using Lemmas 26, 29, 32, and 35, get the following result.

▶ **Lemma 36.** *If the Exponential Time Hypothesis (ETH) holds then there exists a constant $c > 1$ such that any c-approximation algorithm for the k-means problem cannot have running time better than $poly(n, d) \cdot 2^{\Omega(\frac{k}{poly \log k})}$.*

This proves Theorem 6 since if there is a query algorithm that runs in time $poly(n, d, k)$ and makes $\frac{k}{poly \log k}$ same-cluster queries, then we can convert it to a non-query algorithm that runs in time $poly(n, d) \cdot 2^{\frac{k}{poly \log k}}$ in a brute-force manner by trying out all possible answers for the queries and then picking the best k-means solution.

5 Query Approximation Algorithm with Faulty Oracle

In this section, we describe how to extend our approximation algorithm for k-means clustering in the SSAC framework when the query oracle is *faulty*. That is, the answers to the same-cluster queries may be incorrect. Let us denote the faulty same-cluster oracle as $\mathcal{O}^E$. We consider the following error model: for a query with points u and v, the query answer $\mathcal{O}^E(u, v)$ is wrong independently with probability at most q where q is a constant strictly less than $1/2$. More specifically, if u and v belong to the same optimal cluster, then $\mathcal{O}^E(u, v) = 1$ with probability at least $(1 - q)$ and $\mathcal{O}^E(u, v) = 0$ with probability at most q. Similarly, if u and v belong to different optimal clusters, then $\mathcal{O}^E(u, v) = 1$ with probability at most q and $\mathcal{O}^E(u, v) = 0$ with probability at least $(1 - q)$.

The modified algorithm giving $(1 + \varepsilon)$-approximation for k-means with faulty oracle $\mathcal{O}^E$ is given in Table 3. Let $X_1, \ldots, X_k$ denote the k optimal clusters for the dataset X. Let $C = \{c_1, \ldots, c_i\}$ denote the set of i centers chosen by the algorithm at the end of iteration i. Let S denote the sample obtained using D^2-sampling in the $(i + 1)^{\text{st}}$ iteration. The key idea for an efficient $(1 + \varepsilon)$-approximation algorithm for k-means in the SSAC framework with a *perfect* oracle was the following. Given a sample S, we could compute using at most $k|S|$ same-cluster queries the partition $S_1, \ldots, S_k$ of S among the k optimal clusters such that $S_j = S \cap X_j$ for all j. In the following, we discuss how to achieve this partitioning of S among k optimal clusters when the oracle $\mathcal{O}^E$ is faulty.

We reduce the problem of finding the partitions of S among the optimal clusters to the problem of recovering dense (graph) clusters in a *stochastic block model* (SBM). An instance of an SBM is created as follows. Given any arbitrary partition $V_1, \ldots, V_k$ of a set V of vertices, an edge is added between two vertices belonging to the same partition with probability at least $(1 - q)$ and between two vertices in different partitions with probability at most q. We first construct an instance I of an SBM using the sample S. By querying the oracle $\mathcal{O}^E$ with all pairs of points u, v in S, we obtain a graph I on $|S|$ vertices, where vertices in I correspond to the points in S, and an edge exists in I between vertices u and v if $\mathcal{O}^E(u, v) = 1$. Since oracle $\mathcal{O}^E$ errs with probability at most q, for any $u, v \in S_j$ for some $j \in [k]$, there is an edge between u and v with probability at least $(1 - q)$. Similarly, there is an edge $(u, v) \in I$ for any two points $u \in S_y$ and $v \in S_z, y \neq z$ belonging to different optimal clusters with probability at most q. Note that the instance I created in this manner would be an instance of an SBM. Since $q < 1/2$, this procedure, with high probability, creates more edges between vertices belonging to the same partition than the number of edges between vertices in different partitions. Intuitively, the partitions of S would correspond to the dense (graph) clusters in the SBM instance I, and if there were no errors, then each partition would form a clique in I. One way to figure out the partitions $S_1, \ldots, S_k$ would be to retrieve the

Table 3 Approximation algorithm for k-means (top-left frame) using faulty oracle. Note that $\mu(T)$ denotes the centroid of T and D^2-sampling w.r.t. empty center set C means just uniform sampling. The algorithm `UniformSample`(X, C, s) (bottom-left) returns a uniform sample of size $\Omega(1/\varepsilon)$ (w.h.p.) from the optimal cluster in which point s belongs.

Constants: $N = \frac{(2^{13})k^3}{\varepsilon^2}$, $M = \frac{64k}{\varepsilon}$, $L = \frac{(2^{23})k^2}{\varepsilon^4}$

`Faulty-Query-`k`-means`(X, k, ε)

(1) $R \leftarrow \emptyset$

(2) $C \leftarrow \emptyset$

(3) for $i = 1$ to k

(3.1) D^2-sample a multi-set S of N points from X with respect to center set C

(3.2) $s \leftarrow$ `UncoveredCluster`(C, S, R)

(3.3) $T \leftarrow$ `UniformSample`(X, C, s)

(3.4) If $(|T| < M)$ continue

(3.5) $R \leftarrow R \cup \{s\}$

(3.6) $C \leftarrow C \cup \mu(T)$

(4) return(C)

`UncoveredCluster`(C, S, R)

- For all $i \in \{1, ..., k\}$: $S_i \leftarrow \emptyset$
- $i \leftarrow 1$
- For all $y \in R$: $\{S_i \leftarrow y$; i++$\}$
- $T_1, \ldots, T_l =$ `PartitionSample`(S)
- for $j = 1, \ldots, l$
  - if IsCovered(C, T_j) is FALSE
    - if $\exists t$ such that $S_t = \emptyset$, then $S_t = T_j$
- Let S_i be the largest set such that $i > |R|$
- Let $s \in S_i$ be the element with smallest value of $\Phi(C, \{s\})$ in S_i
- return(s)

`UniformSample`(X, C, s)

- $S \leftarrow \emptyset$
- For $i = 1$ to L:
  - D^2-sample point $x \in X$ w.r.t center set C
  - $U = U \cup \{x\}$
- $T_1, \ldots, T_l =$ `PartitionSample`(U)
- for $j = 1, \ldots, l$
  - If (`IsCovered`(s, T_j) is TRUE)
    - $\forall x \in T_j$, with probability $\left(\frac{\varepsilon}{128} \cdot \frac{\Phi(C,\{s\})}{\Phi(C,\{x\})}\right)$ add x in multi-set S
- return (S)

`PartitionSample`(S)

- Construct SBM instance I by querying $\mathcal{O}^E(s, t)$ $\forall s, t \in S$
- Run cluster recovery algorithm of Ailon et al. [4] on I
- Return $T_1, \ldots, T_l$ for $l < k$

`IsCovered(C,U)`

- for $c \in C$
  - if for majority of $u \in U$, $\mathcal{O}^E(c, u) = 1$
    - Return TRUE
- Return FALSE

dense (graph) clusters from the instance I. Ailon et al. [4] gave a randomized algorithm to retrieve all large clusters of any SBM instance. Their algorithm on a graph of n vertices retrieves all clusters of size at least $\sqrt{n}$ with high probability. Their main result in our context is given as follows.

▶ **Lemma 37** ([4]). *There exists a polynomial time algorithm that, given an instance of a stochastic block model on n vertices, retrieves all clusters of size t least $\Omega(\sqrt{n})$ with high probability, provided $q < 1/2$.*

We use Lemma 37 to retrieve the large clusters from our SBM instance I. We also need to make sure that the sample S is such that its overlap with at least one uncovered optimal

cluster is large, where an optimal cluster S_j for some j is *uncovered* if $C \cap S_j = \emptyset$. More formally, we would require the following: $\exists j \in [k]$ such that $|S_j| \geq \Omega(\sqrt{|S|})$, and X_j is uncovered by C. From Corollary 16, given a set of centers C with $|C| < k$, there exists an uncovered cluster such that any point sampled using D^2-sampling would belong to that uncovered cluster with probability at least $\frac{\varepsilon}{4k}$. Therefore, in expectation, D^2-sample S would contain at least $\frac{\varepsilon}{4k}|S|$ points from one such uncovered optimal cluster. In order to ensure that this quantity is at least as large as $\sqrt{|S|}$, we need $|S| = \Omega(\frac{16k^2}{\varepsilon^2})$. Our bounds for N and L, in the algorithm, for the size of D^2-sample S satisfy this requirement with high probability. This follows from a simple application of Chernoff bounds.

▶ **Lemma 38.** *For D^2-sample S of size at least $\frac{2^{12}k^2}{\varepsilon^2}$, there is at least one partition $S_j = S \cap X_j$ among the partitions returned by the sub-routine* `PartitionSample` *corresponding to an uncovered cluster X_j with probability at least $(1 - \frac{1}{16k})$.*

Proof. From Corollary 16, for any point p sampled using D^2-sampling, the probability that point p belongs to some uncovered cluster X_j is at least $\frac{\varepsilon}{4k}$. In expectation, the number of points sampled from uncovered cluster X_j is $\mathbf{E}[|S_j|] = \frac{\varepsilon|S|}{4k} = \frac{2^{10}k}{\varepsilon}$. Exact recovery using Lemma 37 requires $|S_j|$ to be at least $\frac{2^6 k}{\varepsilon}$. Using Chernoff bounds, the probability of this event is at least $(1 - \frac{1}{16k})$. ◀

Following Lemma 38, we condition on the event that there is at least one partition corresponding to an uncovered cluster among the partitions returned by the sub-routine `PartitionSample`. Next, we figure out using the sub-routine `IsCovered` which of the partitions returned by `PartitionSample` are covered and which are uncovered. Let $T_1, \ldots, T_l$ be the partitions returned by `PartitionSample` where $l < k$. Sub-routine `IsCovered` determines whether a cluster is covered or uncovered in the following manner. For each $j \in [l]$, we check whether T_j is covered by some $c \in C$. We query oracle $\mathcal{O}^E$ with pairs (v, c) for $v \in T_j$ and $c \in C$. If majority of the query answers for some $c \in C$ is 1, we say cluster T_j is covered by C. If for all $c \in C$ and some T_j, the majority of the query answers is 0, then we say T_j is uncovered by C. Using Chernoff bounds, we show that with high probability uncovered clusters would be detected.

▶ **Lemma 39.** *With probability at least $(1 - \frac{1}{16k})$, all covered and uncovered clusters are detected correctly by the sub-routine* `IsCovered`*.*

Proof. We find the probability that any partition T_j for $j \in [l]$ is detected correctly as covered or uncovered. Then we use union bound to bound the probability that all clusters are detected correctly. Recall that each partition returned by `PartitionSample` has size at least $|T_j| \geq \frac{2^6 k}{\varepsilon}$ for $j \in [l]$. We first compute for one such partition T_j and some center $c \in C$, the probability that majority of the queries $\mathcal{O}^E(v, c)$ where $v \in T_j$ are wrong. Since each query answer is wrong independently with probability $q < 1/2$, in expectation the number of wrong query answers would be $q|T_j|$. Using Chernoff bound, the probability that majority of the queries is wrong is at most $e^{-\frac{2^6 k}{3\varepsilon}(1-\frac{1}{2q})^2}$. The probability that the majority of the queries is wrong for at least one center $c \in C$ is at most $ke^{-\frac{2^6 k}{3\varepsilon}(1-\frac{1}{2q})^2}$. Again using union bound all clusters are detected correctly with probability at least $(1 - k^2 e^{-\frac{2^6 k}{3\varepsilon}(1-\frac{1}{2q})^2}) \geq (1 - \frac{1}{16k})$. ◀

With probability at least $(1 - \frac{1}{8k})$, given a D^2-sample S, we can figure out the largest uncovered optimal cluster using the sub-routines `PartitionSample` and `IsCovered`. The analysis of Algorithm 3 follows the analysis of Algorithm 2. For completeness, we compute the probability of success, and the query complexity of the algorithm. Note that s in line (3.2)

of the Algorithm 3 is chosen correctly with probability $(1-\frac{1}{4k})(1-\frac{1}{8k})$. The uniform sample in line (3.3) is chosen properly with probability $(1-\frac{1}{4k})(1-\frac{1}{8k})$. Since, given the uniform sample, success probability using Inaba's lemma is at least $(1-\frac{1}{4k})$, overall the probability of success becomes $(1-\frac{1}{k})$. For query complexity, we observe that `PartitionSample` makes $O(\frac{k^6}{\varepsilon^8})$ same-cluster queries to oracle $\mathcal{O}^E$, query complexity of `IsCovered` is $O(\frac{k^4}{\varepsilon^4})$. Since `PartitionSample` is called at most k times, total query complexity would be $O(\frac{k^7}{\varepsilon^8})$. Note that these are bounds for dataset that satisfies (k,ε)-irreducibility condition. For general dataset, we will use $O(\varepsilon/k)$ as the error parameter. This causes the number of same-cluster queries to be $O(k^{15}/\varepsilon^8)$. This proves the main result in Theorem 7.

6 Conclusion and Open Problems

This work explored the power of the SSAC framework defined by Ashtiani *et al.* [6] in the approximation algorithms domain. We showed how a simple modification of k-means++ seeding algorithm in SSAC framework gives a constant factor approximation for k-means. Furthermore, we obtained an efficient $(1+\varepsilon)$-approximation algorithm for the k-means problem within the SSAC framework. This is interesting because it is known that such an efficient algorithm is not possible in the classical model unless $\mathsf{P}=\mathsf{NP}$.

Our results encourage us to formulate similar query models for other hard problems. If the query model is reasonable (as is the SSAC framework for center-based clustering), then it may be worthwhile to explore its powers and limitations as it may be another way of circumventing the hardness of the problem. For instance, the problem closest to center-based clustering problems such as k-means is the *correlation clustering* problem. The query model for this problem may be similar to the SSAC framework. It will be interesting to see if same-cluster queries allow us to design efficient approximation algorithms for correlation clustering problem for which hardness results similar to that of k-means are known.

References

1 Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *APPROX-RANDOM*, pages 15–28. Springer, 2009.

2 Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *CoRR*, abs/1612.07925, 2016. `arXiv:1612.07925`.

3 Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. *CoRR*, abs/1704.01862, 2017. `arXiv:1704.01862`.

4 Nir Ailon, Yudong Chen, and Huan Xu. Iterative and active graph clustering using trace norm minimization without cluster size constraints. *Journal of Machine Learning Research*, 16:455–490, 2015.

5 David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

6 Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. In *Advances in neural information processing systems*, pages 3216–3224, 2016.

7 Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. In *ICML*, pages 550–558, 2014.

8 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012.

9 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34, pages 754–767, 2015.

10 Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In *ALT*, pages 316–328. Springer, 2008.

11 Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1068–1077, 2009.

12 Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM, 2004.

13 Anup Bhattacharya, Ragesh Jaiswal, and Nir Ailon. Tight lower bound instances for k-means++ in two dimensions. *Theoretical Computer Science*, 634:55–66, 2016.

14 Tobias Brunsch and Heiko Röglin. A bad instance for k-means++. *Theoretical Computer Science*, 505:19–26, 2013.

15 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 353–364. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.46.

16 Sanjoy Dasgupta. The hardness of k-means clustering. Technical report, Department of Computer Science and Engineering, University of California, San Diego, 2008.

17 Irit Dinur. The pcp theorem by gap amplification. *Journal of the ACM (JACM)*, 54(3):12, 2007.

18 Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A ptas for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18. ACM, 2007.

19 Zachary Friggstad, Mohsen Rezapour, and Mohammad R Salavatipour. Local search yields a ptas for k-means in doubling metrics. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 365–374. IEEE, 2016.

20 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62:367–375, 2001.

21 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512–530, 2001.

22 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 332–339. ACM, 1994.

23 Ragesh Jaiswal, Amit Kumar, and Sandeep Sen. A simple d2-sampling based ptas for k-means and other clustering problems. *Algorithmica*, 70(1):22–46, 2014.

24 Ragesh Jaiswal, Mehul Kumar, and Pulkit Yadav. Improved analysis of d^2-sampling based ptas for k-means and other clustering problems. *Information Processing Letters*, 115(2):100–103, 2015.

25 Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18. ACM, 2002.

26 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *Journal of the ACM (JACM)*, 57(2):5, 2010.

27 Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

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

29 Pasin Manurangsi. Almost-polynomial ratio eth-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–961. ACM, 2017.

30 Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):28, 2012.

31 Luca Trevisan. Inapproximability of combinatorial optimization problems. *CoRR*, cs.CC/0409043, 2004. URL: `http://arxiv.org/abs/cs.CC/0409043`.

32 Andrea Vattani. The hardness of k-means clustering in the plane. Technical report, Department of Computer Science and Engineering, University of California San Diego, 2009.

33 Sharad Vikram and Sanjoy Dasgupta. Interactive bayesian hierarchical clustering. In *International Conference on Machine Learning*, pages 2081–2090, 2016.

34 Konstantin Voevodski, Maria-Florina Balcan, Heiko Röglin, Shang-Hua Teng, and Yu Xia. Efficient clustering with limited distance information. *CoRR*, abs/1408.2045, 2014. `arXiv:1408.2045`.