Improved Algorithms for Clustering with Noisy Distance Oracles

Pinki Pradhan
NISER, Bhubaneswar
An OCC of HBNI, India

Anup Bhattacharya
NISER, Bhubaneswar
An OCC of HBNI, India

Ragesh Jaiswal
IIT Delhi, India

Abstract

Bateni *et al.* has recently introduced the *weak-strong distance oracle model* to study clustering problems in settings with limited distance information. Given query access to the strong-oracle and weak-oracle in the weak-strong oracle model, the authors design approximation algorithms for k-means and k-center clustering problems. In this work, we design algorithms with improved guarantees for k-means and k-center clustering problems in the weak-strong oracle model. The k-means++ algorithm is routinely used to solve k-means in settings where complete distance information is available. One of the main contributions of this work is to show that k-means++ algorithm can be adapted to work in the weak-strong oracle model using only a small number of strong-oracle queries, which is the critical resource in this model. In particular, our k-means++ based algorithm gives a constant approximation for k-means and uses $O(k^2 \log^2 n)$ strong-oracle queries. This improves on the algorithm of Bateni *et al.* that uses $O(k^2 \log^4 n \log^2 \log n)$ strong-oracle queries for a constant factor approximation of k-means. For the k-center problem, we give a simple *ball-carving* based $6(1+\epsilon)$-approximation algorithm that uses $O(k^3 \log^2 n \log \frac{\log n}{\epsilon})$ strong-oracle queries. This is an improvement over the $14(1+\epsilon)$-approximation algorithm of Bateni *et al.* that uses $O(k^2 \log^4 n \log^2 \frac{\log n}{\epsilon})$ strong-oracle queries. To show the effectiveness of our algorithms, we perform empirical evaluations on real-world datasets and show that our algorithms significantly outperform the algorithms of Bateni *et al.*

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 INTRODUCTION

Clustering problems such as the k-means, k-median and k-center are often studied in *complete distance information* settings where the embeddings of the n points to be clustered are given as input to the problem. These k-clustering problems are commonly formulated to minimize some cost function of the input. Often these formulations turn out to be NP-hard, and approximation algorithms are designed to solve these problems. However, for many machine learning applications, the assumption of complete knowledge of the dataset under consideration might be either infeasible or very expensive to meet. This motivates the study of clustering problems in *partial distance information* settings, where there is a tradeoff between the accuracy of the input data and the resulting cost of clustering. The more accurate you want the clustering solution to be, the cost of collecting accurate information about the dataset would become higher. Questions such as *what is the best computational guarantee (e.g., approximation factor) that one can obtain within a budget*, or *what is the minimum cost for obtaining a target guarantee*, become meaningful. Such studies can also be interpreted as computing with noisy information and are not entirely new (e.g., Feige et al. (1994)), and the problems considered in these settings range from sorting and searching to graph problems (Gu and Xu, 2023; Karp and Kleinberg, 2007).

Clustering problems have been studied in various partial information settings. Often these works are formulated as having access to an oracle that provides some meaningful information about the underlying clustering, and the goal is to obtain a good approximate clustering solution using only a small number of oracle queries. Several works studied variants of this problem with different assumptions about the oracle and the clustering objective. Liu and Mukherjee (2022) give tight query complexity bounds

for the cluster recovery problem with a membership-query oracle, which on a query with any two points, answers whether the points belong to the same cluster or not. Several works (Ailon et al., 2018; Mazumdar and Saha, 2017) have studied clustering using a noisy membership-query oracle, also called a same-cluster query oracle, to get approximate solutions. Gamlath et al. (2022) study algorithms for k-means where noisy labels are provided for each input point using an adversarial or random perturbation of the labels.

In this work, we study clustering problems in the *weak-strong oracle model* introduced by Bateni et al. (2024). The weak-strong oracle model has access to an expensive strong-oracle giving exact distances between any two points and a cheaper weak-oracle that gives noisy answers for the distances. Given that meaningful vector embeddings of varied accuracy are now frequently available in most machine learning applications for various types of data, it makes sense to consider a noise model where the distances (indicating dissimilarity) between data items are available, and accurate information comes at a cost. This was the primary motivation behind the weak-strong oracle model given by Bateni et al. (2024). Next, we describe clustering problems with access to distance oracles.

In the clustering with distance oracles problem, the algorithm is given as input the number n of points in the dataset and the number k of clusters. We assume that the input points X belong to some metric space $(\mathcal{X}, d)$. However, we don't have direct access to their embeddings, and also we don't know the distances between any two of these points. We are given query access to distance oracles that, on a query with any two points in $\mathcal{X}$, respond with the distance between the two points. Each query has a cost associated with it, and we assume the strong-oracle queries to be much more expensive compared to the weak-oracle queries. We design algorithms for clustering problems that make only a small number of queries to these distance oracles and return good approximate solutions. The first distance oracle we mention is the strong-oracle SO which returns the exact distance $d(x, y)$ for any two query points $x, y \in \mathcal{X}$.

Definition 1.1 (Strong-oracle model)**.** *In this model, an algorithm is given access to a strong-oracle that for any $x, y \in \mathcal{X}$, returns $SO(x, y) = d(x, y)$, the exact distance between the two points.*

We first observe that $O(nk)$ strong-oracle queries suffice to run the k-means++ algorithm, and using known results, one gets a constant factor bi-criteria approximation for k-means (Aggarwal et al., 2009) using $O(nk)$ strong-oracle queries. We also observe that $O(nk)$ strong-oracle queries are enough for a 2-approximation for k-center problem (Gonzalez, 1985). However, as mentioned earlier, access to such strong-oracles might be limited and quite expensive in practice. This motivates us to design algorithms that access a much cheaper and possibly erroneous weak-oracle $\widetilde{WO}$. For some fixed $\delta \in (0, 1/2)$, given two points $x, y \in \mathcal{X}$, the weak-oracle $\widetilde{WO}$ returns the exact distance $d(x, y)$ between the two points independently with probability $(1 - \delta)$ and returns an arbitrary answer with the remaining probability. Moreover, we also assume that the query answers of the weak-oracle $\widetilde{WO}$ are *persistent* in the sense that the weak-oracle always returns the same answer to a query, even if asked multiple times.

Definition 1.2 (Weak-oracle model)**.** *In this model, an algorithm has access to a weak-oracle $\widetilde{WO}$ that for any two points $x, y \in \mathcal{X}$, outputs $d(x, y)$ with probability $1 - \delta$, and with remaining probability δ, outputs an arbitrary value.*

Weak-strong oracle model of Bateni et al. (2024): An algorithm in this model is allowed to query both the weak-oracle and the strong-oracle. We show that algorithms with good guarantees can be designed for clustering problems in this model that use only a small number of strong-oracle queries.

We note here that the assumption on the weak-oracle that it returns the exact distance between any two points with constant probability might appear to be quite *strong* for applications in practice. In Appendix D, we give some informal justification for the weak-strong oracle model arguing that some natural attempts to model noisy oracle distances can be thought of as instantiations of the weak-oracle model.

1.1 Clustering Problems

Let X be a set of n points in a metric space $(\mathcal{X}, d)$. The k-means cost of X with respect to any set $C \subset \mathcal{X}$ is given as $\Phi(X, C) = \sum_{x \in X} d^2(x, C)$, where $d(x, C) = \min_{c \in C} d(x, c)$. The k-means clustering problem is defined as follows: *Given a set X of n points in a metric space $(\mathcal{X}, d)$ and an integer k, the objective in the k-means problem is to output a set C of k centers for which the k-means cost function $\Phi(X, C)$ is minimized.* The k-means clustering problem is very well studied theoretically and has numerous applications in practice. This problem is known to be NP-hard (Dasgupta, 2008). The k-means++ algorithm is often used to solve k-means in practice (Arthur and Vassilvitskii, 2007). The main idea is to iteratively sample k points using a distribution called D^2-distribution that is updated after every iteration and depends on the previously chosen centers. It is known that k-means++ gives $O(\log k)$-approximation in expectation for k-means (Arthur and Vassilvitskii,

2007). Moreover, it can also be adapted to obtain a constant factor bi-criteria approximation (Aggarwal et al., 2009) simply by oversampling centers (i.e., sample more than k centers). We shall refer to this as oversampling-k-means++. An (α, β)-bi-criteria approximation uses at most αk centers and returns a solution of cost at most β times the optimal k-means solution. The (k, z)-clustering problem is a generalization of the k-means problem. In the (k, z)-clustering problem, given a set X of n points from a metric space and an integer k, the objective is to return a set $C \subset X$ of k centers for which $\sum_{x \in X} d(x, C)^z$ is minimized. For $z = 2$, this is the k-means problem and for $z = 1$, this is the k-median problem.

Another way to formulate data clustering is using the k-center problem, where the goal is to optimize the following cost function. The k-center cost of X with respect to a set $C \subset \mathcal{X}$ is defined as $\phi(X, C) = \max_{x \in X} d(x, C)$, where $d(x, C) = \min_{c \in C} d(x, c)$. More specifically, the k-center problem is defined as follows: *Given a set X of n points in a metric space $(\mathcal{X}, d)$ and an integer k, the objective in the k-center problem is to output a set C of k centers for which the k-center cost function $\phi(X, C)$ is minimized.* In other words, the goal is to find k points such that the radius of the balls centered at these points that cover X, is minimized. The k-center problem is known to be NP-hard and 2-approximation algorithms are known for the k-center problem (Gonzalez, 1985).

1.2 Main Results

Weak-strong oracle model: We design algorithms for k-means and k-center problems in the weak-strong oracle model that use only a small number of strong-oracle queries while giving good approximate solutions. Our algorithms provide better guarantees compared to the work of Bateni et al. (2024). We next state our main results for k-means and k-center problems, and provide more details in Sections 2 and 3, respectively.

Theorem 1.1. *(Upper bound for k-means) Let $\epsilon \in (0, 1)$ and $\delta \leq 1/3$. There exists a randomized algorithm for the k-means problem that adapts oversampling k-means++ in the weak–strong oracle model and outputs a $\left(O\left(\frac{\log n}{\epsilon^4}\right), 40(1+\epsilon)\right)$ bi-criteria approximation for k-means. The algorithm uses $O\left(\frac{k^2 \log^2 n}{\epsilon^8}\right)$ strong-oracle queries and $O\left(\frac{nk \log n}{\epsilon^4}\right)$ weak-oracle queries, and succeeds with constant probability.*

Theorem 1.2. *(Upper bound for k-center) Let $\epsilon > 0$. There exists a $6(1+\epsilon)$-approximation algorithm for the k-center problem that makes $O\left(k^3 \log^2 n \log \frac{\log n}{\epsilon}\right)$ strong-oracle and $O\left(nk \log n \log \frac{\log n}{\epsilon}\right)$ weak-oracle queries in the weak-strong oracle model and succeeds with probability at least $(1 - 1/n^4)^2$.*

Remark 1.1. *We obtain these results for the failure probability $\delta = 1/3$. We note that our algorithms can be extended to work with any $0 < \delta < 1/2$. The dependency of δ on the sample complexity would become $1/(1/2 - \delta)^2$ Bateni et al. (2024).*

Weak-oracle model: An algorithm in this query model can only access the weak-oracle $\widetilde{WO}$. We show a lower bound result in the weak-oracle model proving that any constant factor approximate clustering solution for k-means in this model requires $\Omega(nk/(1-2\delta)^2)$ weak-oracle queries. We prove the following result in Appendix G.

Theorem 1.3. *Let $\delta \in (0, 1/2)$. Any randomized algorithm giving a $O(1)$-approximation for k-means in weak-oracle model with probability at least $3/4$ requires $\Omega(nk/(1-2\delta)^2)$ queries in expectation.*

Our contributions and comparisons with known works In this work, we provide improved algorithms for clustering problems in the weak-strong oracle model of Bateni et al. (2024). We highlight the main contributions of our work and provide a detailed comparison of our results with Bateni et al. (2024).

- k-means++ algorithm is widely used to solve k-means in practice. k-means++ algorithm has also been extended to work in the presence of outliers (Bhaskara et al., 2019; Deshpande and Pratap, 2023), and even when there are errors in the computation of the D^2-sampling distribution (Bhattacharya et al., 2020; Grunau et al., 2023). However, it was not known whether k-means++ works with unreliable distance estimates as in the weak-strong model. We show that even though the distance estimates given by the weak-oracle are unreliable, it is possible to come up with an alternate distance measure with respect to which one can run the k-means++ algorithm to obtain a good approximation guarantee for k-means. We believe that one of the main contributions of this work is to adapt the k-means++ algorithm to work in this restricted model.
- Bateni et al. (2024) give a constant approximation algorithm for k-means using $O(k^2 \log^4 n \log^2(\log n/\epsilon))$ strong-oracle queries and $O(nk \log^2 n \log(\log n/\epsilon))$ weak-oracle queries. Our algorithm for k-means uses $O(\frac{k^2 \log^2 n}{\epsilon^8})$ strong-oracle queries and $O(\frac{nk \log n}{\epsilon^4})$ weak-oracle queries and gives a constant approximation for k-means, with a better constant factor.

Table 1: Comparison of approximation guarantees and query complexities for k-means and k-center problems with Bateni et al. (2024) in the weak-strong oracle model. For k-means, we report the bi-criteria approximation guarantees obtained by our algorithm as well as Bateni et al. (2024).

Problems	Results	Approximation Guarantee	Strong-oracle Queries	Weak-oracle Queries
k-means	Bateni et al. (2024)	$(O(\log^2 n \log(\frac{\log n}{\epsilon})), O(1))$[1]	$O(k^2 \log^4 n \log^2(\frac{\log n}{\epsilon}))$	$O(nk \log^2 n \log(\frac{\log n}{\epsilon}))$
	Our	$(O(\frac{\log n}{\epsilon^4}), 40(1+\epsilon))$	$O\left(\frac{k^2 \log^2 n}{\epsilon^8}\right)$	$O\left(\frac{nk \log n}{\epsilon^4}\right)$
k-center	Bateni et al. (2024)	$14(1+\epsilon)$	$O(k^2 \log^4 n \log^2(\frac{\log n}{\epsilon}))$	$O(nk \log^2 n \log(\frac{\log n}{\epsilon}))$
	Our	$6(1+\epsilon)$	$O\left(k^3 \log^2 n \log\left(\frac{\log n}{\epsilon}\right)\right)$	$O\left(nk \log n \log\left(\frac{\log n}{\epsilon}\right)\right)$

- Our algorithm for k-means in the weak-strong oracle model can be generalized to obtain a $O(2^{2z})$ factor approximation in expectation for the (k, z) clustering problem.

- Bateni et al. (2024) adapts Meyerson's sketch for online facility location for k-means problem in the weak-strong model Meyerson (2001). Their algorithm guesses the optimal value of k-means solution, and using a bounded aspect-ratio assumption on the input instance, they show that the number of such guesses is limited. Our algorithm doesn't use any such assumption.

- For the k-center problem, we obtain the following improvements over Bateni et al. (2024). We give a $6(1+\epsilon)$-approximation for k-center using $O(k^3 \log^2 n \log\log n/\epsilon)$ strong-oracle queries and $O(nk \log n \log\log n/\epsilon)$ weak-oracle queries. This is in contrast to Bateni et al. (2024) who give a $14(1+\epsilon)$-approximation using $O(k^2 \log^4 n \log^2(\log n/\epsilon))$ strong-oracle and $O(nk \log^2 n \log(\log n/\epsilon))$ weak-oracle queries.

Experiments We experimentally verify the performance of our algorithms for k-means and k-center problems on synthetic as well as real-world datasets. For experiments on synthetic datasets, we use datasets generated using the stochastic block model (Mossel et al., 2015; Abbe, 2018). For experiments on real-world datasets, we use MNIST dataset Deng (2012) with SVD and t-SNE embeddings (Van der Maaten and Hinton, 2008). We compare the experimental results of our algorithms with those of Bateni et al. (2024), and note that our algorithms use significantly fewer number of queries to output clusterings of comparable cost. Our algorithms for k-means and k-center use at least 61% and 23% fewer queries, respectively, compared to Bateni et al. (2024). The experimental details are provided in Section 4 and Appendix I.

[1]Bateni et al. (2024) only mention a $O(1)$-approximation factor. We estimate the constant factor to be much larger than 40.

1.3 Technical Overview

In this section, we describe the main ideas of our algorithm for k-means in the weak-strong oracle model. One of the most widely used algorithms for k-means is the k-means++ seeding algorithm that works in k iterations. In the first iteration, it chooses the first center as a point of the input sampled uniformly at random. Each of the remaining $(k-1)$ iterations samples a point following a non-uniform and adaptive sampling distribution known as the D^2 distribution that depends on the previously chosen centers. Sampling a point from the D^2-distribution is called D^2-sampling. The k points sampled using D^2-sampling are returned as the solution for k-means. The D^2-distribution is such that the probability of choosing a point x is proportional to the squared distance $d(x, C)^2$, where C is a set of chosen centers and $d(x, C) = \min_{c \in C} d(x, c)$. Aggarwal et al. (2009) showed a constant factor bi-criteria approximation for the oversampling version of k-means++ in which $O(k)$ iterations of D^2-sampling are used.

One of the main contributions of this work is to show that one can adapt k-means++ to the weak-strong oracle model. It is easy to observe that one can simulate k-means++ in the strong-oracle model using $O(nk)$ strong-oracle queries. However, things become complicated in the weak-strong model, in which the distances returned by the weak-oracle cannot be trusted. To construct the D^2-sampling distribution, one must calculate $d(x, C)$ for any point x and any set C of centers. Since $d(x, C) = \min_{c \in C} d(x, c)$, if we want to compute this distribution exactly, it would require $\Omega(nk)$ strong-oracle queries. Since we don't want to use too many strong-oracle queries, we proceed as follows. Using weak-oracle query answers, we compute an estimate $d_{km}^{est}(x, C)$ such that with high probability, $d(x, C)$ and $d_{km}^{est}(x, C)$ are within some small additive factor. We construct the sampling distribution to be used by the algorithm using these $d_{km}^{est}(x, C)$ values for all points x.

To compute $d_{km}^{est}(x, C)$ for all $x \in X$, we use the fact that weak-oracle distance queries are incorrect

independently with probability $\delta < 1/2$. Consider any $c \in C$. Let us obtain an estimate on $d(x, c)$ using only weak-oracle queries. Let $B(c, r_c)$ denote a ball around the center c of radius r_c, and suppose that r_c is chosen such that $B(c, r_c)$ contains $\Omega(\log n)$ points. Then, the key idea is that the median of the weak-oracle distance queries on pairs (x, y) with $y \in B(c, r_c)$ is a good estimate on $d(x, c)$. More formally, we use $d^{est}_{km}(x, c) = median\{\widetilde{WO}(x, y) : y \in B(c, r_c)\}$ as an estimate for $d(x, c)$. We note that Bateni et al. (2024) also uses the median of query answers to estimate $d(x, c)$. However, their algorithm used these values differently, as we will see in the remaining discussion. Since each weak-oracle query answer is wrong independently with probability at most 1/2, using Chernoff bounds, one can show that for sufficiently large-sized ball $B(c, r_c)$, with high probability, $|d(x, c) - d^{est}_{km}(x, c)| \leq r_c$. Now, to approximate $d(x, C)$, we compute $d^{est}_{km}(x, c) + r_c$ for all $c \in C$ and set $d^{est}_{km}(x, C) = \min_{c \in C}\{d^{est}_{km}(x, c) + r_c\}$. Once we have the sampling distribution $\widetilde{D^2}$ for which a point x is sampled with probability proportional to $d^{est}_{km}(x, C)^2$, we show that the approach of Aggarwal et al. (2009) can be adapted to obtain a constant factor bi-criteria approximation for k-means that succeeds with constant probability. Even though the analysis becomes non-trivial and we incur an approximation loss, the high-level analysis ideas of Aggarwal et al. (2009) goes through.

The main idea is to run the oversampling k-means++ where we iteratively sample and update the center set C. However, the sampling is with respect to $d^{est}_{km}(x, C)^2$ instead of $d(x, C)^2$. The goal is to pick a good center from every optimal cluster. Let us see why oversampling using $d^{est}_{km}(x, C)^2$ will be sufficient to find good centers from every optimal cluster, and hence obtain a good approximation guarantee. For the simplicity of discussion, let us first assume that every optimal cluster has $\Omega(\log n)$ points. We will later see how to drop this assumption. Let us draw a parallel with oversampling k-means++ that samples using accurate distance values. Why does it manage to pick good centers from every optimal cluster? Consider an intermediate center set C. Certain optimal clusters will have a good representative center in C, whereas others remain 'uncovered'. Sampling using the D^2 distribution boosts the probability of sampling from an uncovered cluster, so there is a good chance that the next sample belongs to an uncovered cluster. Moreover, we can argue that given the next sample is from an uncovered cluster, there is a good chance that it will be a good center from that cluster.

Let's see whether the same argument applies when sampling uses $d^{est}_{km}(x, C)^2$ instead of $d(x, C)^2$. If our center set C has a single good center (or a few) from an optimal cluster, say A, can we consider the cluster to be covered? No. The probability of sampling from A may remain high since the distance estimate of points in A to the centers in $C \cap A$ may be completely off. When does the distance estimate start becoming tighter? This happens when there are $\Omega(\log n)$ good centers from A in C. This is when we can call the cluster A 'settled'. So, instead of 'covering' every cluster (if correct distances are known), we care about 'settling' every cluster in our current model. So, let us see if we can settle every cluster if we keep sampling centers. We can show that unless the current center set C is already good, there is a good chance that the next center will be sampled from one of the unsettled optimal clusters. Further, we can also argue that conditioned on sampling from an unsettled cluster, there is a good chance that the sampled center will be a good center (i.e., reasonably close to the optimal center). So, as long as we sample sufficiently many centers, every optimal cluster will get settled with high probability. The oversampling factor is $O(\log n)$, i.e., we end up with a center set C with $O(k \log n)$ centers. Finally, we will use $d^{est}_{km}(x, C)$ to assign points x to centers and create a weighted point set C (the weight of a point in C is the number of points assigned to it), on which a standard constant approximation algorithm using strong-oracle queries is used to find the final set of k centers. Since the set C has $O(k \log n)$ points, we will need $O(k^2 \log^2 n)$ strong-oracle queries to find all the interpoint distances to run the constant approximation algorithm as well as the distance estimates. Using known techniques, it can be shown that this final center set gives a constant approximation. To drop the assumption that all optimal clusters have $\Omega(\log n)$ points, we argue if an optimal cluster does not have adequate points, the oversampling procedure will sample *all* the points from that cluster with high probability, which is also a favourable case. More discussions on algorithmic ideas for the k-center algorithm and related results can be found in Appendix C.

1.4 Related Works

Bateni et al. (2024) initiate the study of clustering problems in the weak-strong oracle model and design constant approximation algorithms for k-means and k-center problems. They also prove that any constant factor approximation algorithm for k-means or the k-center problem requires $\Omega(k^2)$ strong-oracle queries in the weak-strong model. Galhotra et al. (2024) obtain the following results for clustering problems in a related model. They design constant factor approximation algorithms for k-means and k-center clustering problems given access to a *quadruplet oracle* that takes two pairs (x_1, y_1) and (x_2, y_2) of points as

input and returns whether the pairwise distances are similar or far from each other. A detailed discussion on related work can be found in Appendix B.

2 ALGORITHM FOR K-MEANS IN WEAK-STRONG ORACLE MODEL

In this section we design a constant factor bi-criteria approximate solution for k-means in the weak-strong oracle model with the assumption that all optimal clusters have size at least $480 \log n/\epsilon$. We remove this assumption later. We prove the following result.

Theorem 2.1. *Let $\epsilon \in (0,1)$, $\delta \leq 1/3$. There exists a randomized algorithm for k-means that makes $O(\frac{k^2 \log^2 n}{\epsilon^6})$ strong-oracle queries and $O(\frac{nk \log n}{\epsilon^3})$ weak-oracle queries to give a $\left(O(\frac{\log n}{\epsilon^3}), 40(1+\epsilon)\right)$ bi-criteria approximation for k-means and succeeds with constant probability, assuming every optimal cluster has size at least $\frac{480 \log n}{\epsilon}$.*

Since the weak-oracle gives a wrong answer to a query independently with probability $\delta < 1/2$, we define an alternate distance measure between points and a set of centers that we use in our algorithm. Let C denote a set of centers. For any $x \in X$ and any $c \in C$, we query for the distance between x and c to the weak oracle $\widetilde{WO}$ to obtain a possibly wrong answer $\widetilde{WO}(x,c)$. For any fixed x, we query the weak-oracle for each center c in C, and use these query answers $\widetilde{WO}(x,c)$ to come up with an upper bound on the distance between a point x and a set C of centers defined as follows. We use $\delta \leq 1/3$ for this discussion.

Definition 2.1. *Let x and y be any two points in X and we want to estimate the distance between x and y. Let radius $r_y \geq 0$ be such that the ball $B(y, r_y)$ contains at least $180 \log n$ points. Then, for any $x \in X$, we define the distance from x to y as $d^{est}_{km}(x,y) = median\{\widetilde{WO}(x,z) | z \in B(y, r_y)\}$.*

We make the following claim with respect to the above distance measure. A similar lemma was proved in Bateni et al. (2024). The proof can be found in the Appendix.

Lemma 2.1. *Following Definition 2.1, for any $x \in X$, with probability at least $\left(1 - 1/n^5\right)$, we have $|d^{est}_{km}(x,y) - d(x,y)| \leq r_y$.*

We use the above definition to come up with an upper bound on the distance between a point x and a set C of centers. For any $c \in C$, let r_c denote a radius such that $B(c, r_c)$ has at least $180 \log n$ points inside it. We use Definition 2.1 to come up with an upper bound on the distance between x and any $c \in C$ as $d^{est}_{km}(x,c) + r_c$. Following Lemma 2.1, with high probability, this upper bound holds. That is, with high probability, $d(x,c) \leq d^{est}_{km}(x,c) + r_c$. Using an union bound over all $c \in C$, all these upper bounds hold and hence, the minimum of these upper bounds would also hold. This motivates the following definition on the distance between a point x and a set C of centers.

Algorithm 1: Algorithm for k-means in weak-strong oracle model

Input : Dataset X and an integer $k > 0$, $\epsilon \in (0,1)$, $\delta = 1/3$.
Output: A set of $O(\frac{k \log n}{\epsilon^3})$ centers and an assignment of $x \in X$ to centers.

Let C_1 be a set of $180 \log n$ points chosen arbitrarily from X. `/* Initial centers */`
Set $t = \frac{4320 \cdot 29160}{\epsilon^3} \cdot k \log n$.
for $i = 1$ **to** t **do**
  For each $x \in X$, compute $d^{est}_{km}(x, C_i)$ following Definition 2.2.
  Compute distribution $\widetilde{D^2}$ that samples $x \in X$ with probability proportional to $d^{est}_{km}(x, C_i)^2$.
  Sample a point $s_i \in X$ using distribution $\widetilde{D^2}$.
  `/* Sample new centers following` $\widetilde{D^2}$ `*/`
  Make strong-oracle queries $SO(s_i, c)$ for all $c \in C_i$.
  Update $C_{i+1} \leftarrow C_i \cup \{s_i\}$.
end
Let $\{c_1, c_2, \ldots, c_h\}$ be an arbitrary ordering of the centers in C_{t+1}, where $h = 180 \log n + t$.
Initialize weights $w(c_i) = 0$ for $i \in [h]$.
for $x \in X$ **do**
  Compute $d^{est}_{km}(x, C_{t+1})$ and find c_x for which minimum is achieved in Definition 2.2.
  Assign x to c_x.
  Update $w(c_x) \leftarrow w(c_x) + 1$.
  `/* Construct weighted instance */`
end
return C_{t+1} *and assignment of points in $x \in X$ to centers in C_{t+1} as determined above*

Definition 2.2. *Let x be any point in X and let C be a set of centers of size at least $180 \log n$. For any $c \in C$, let r_c denote a radius value such that the ball $B(c, r_c)$ contains at least $180 \log n$ points. We define the distance between x and C as $d^{est}_{km}(x, C) = \min_{c \in C}\{d^{est}_{km}(x,c) + r_c\}$.*

We state the following lemma with respect to the above distance measure and prove it in Appendix.

Lemma 2.2. *Let x be any point in X and let C denote a set of centers of size at least $180 \log n$. Then, with probability at least $(1 - 1/n^4)$, $d(x, C) \leq d^{est}_{km}(x, C)$. In other words, with probability at least $(1 - 1/n^4)$, there*

exists a center $c \in C$ within a distance of $d_{km}^{est}(x, C)$ from x.

Estimating $d_{km}^{est}(x, C)$ in weak-strong oracle model Next, we describe how we estimate $d_{km}^{est}(x, C)$ for all $x \in X$ in the weak-strong oracle model. We assume that there are at least $180 \log n$ centers in C. Consider any $c \in C$. We query the strong-oracle SO with all pairs $c_i, c_j \in C$ to obtain the exact distances between centers c_i and c_j as $SO(c_i, c_j)$. For each $c \in C$, we find the smallest r_c such that $B(c, r_c)$ contains at least $180 \log n$ points in C. Consider any $x \in X$ for which we want to estimate $d_{km}^{est}(x, C)$. We query the weak-oracle with x and $y \in B(c, r_c)$ to obtain $\widetilde{WO}(x, y)$ for all $y \in B(c, r_c)$, and use these weak-oracle query answers to compute $d_{km}^{est}(x, c)$ following Definition 2.1. We compute $d_{km}^{est}(x, c)$ for all $c \in C$. Finally, to compute $d_{km}^{est}(x, C)$, we find the minimum over all $c \in C$ of $d_{km}^{est}(x, c) + r_c$, as mentioned in Definition 2.2.

We give a high-level description of Algorithm 1. Algorithm 1 starts with a set C of $180 \log n$ centers chosen from X, and computes $d_{km}^{est}(x, C)$ for all x. It constructs the $\widetilde{D^2}$-sampling distribution using which a point $x \in X$ is sampled with probability proportional to $d_{km}^{est}(x, C)^2$. Algorithm 1 samples a center in each of the t iterations and updates the center set C. After t iterations of sampling, it constructs a weighted instance. The solution for k-means is obtained using a constant approximation algorithm on this weighted instance. The detailed analysis of the algorithm is given in Appendix F.

3 ALGORITHM FOR K-CENTER IN WEAK-STRONG ORACLE MODEL

We design a randomized algorithm in the weak-strong oracle model that gives a $6(1+\varepsilon)$-approximation for the k-center problem. We sketch the main idea of the algorithm in a simpler setting where accurate distances are known. We assume that the optimal k-center radius r_{opt} is known. This is not an unreasonable assumption since we can guess the optimal radius within a $(1 \pm \varepsilon)$-factor by iterating over discrete choices of the radius. The algorithm is a simple "*greedy ball-carving*" procedure and can be stated as follows (also stated as Algorithm 3 in Bateni et al. (2024)): *While all points are not removed, pick a center c and remove (carve-out) all points within $2r_{opt}$ of c.* We can show that this procedure outputs at most k centers that is a 2-approximate solution to the k-center problem.

Our $6(1+\varepsilon)$-approximation algorithm follows the above template. The only complication is that during the ball carving step, where we remove all points within the radius $2r_{opt}$, we need accurate distances, which we do not have in the weak-strong oracle model. Let us continue assuming that the optimal radius r_{opt} is known. We will give a 6-approximation under this assumption, which changes to $6(1+\varepsilon)$ when the assumption is dropped. To enable distance estimates, along with a center c_i, we must also pick a set of nearby points to c_i, which will help in estimating the distance of any point y to c_i following Lemma 2.1. A reasonable way to do this would be to uniformly sample a set T_i of points (instead of one), use the strong-oracle queries on the subset and pick a center $c_i \in T_i$ with the largest number of points in T_i within a radius $2r_{opt}$. Let S_{c_i} denote the subset of points in T_i within a distance of $2r_{opt}$ to c_i. We can now obtain distance estimates using the tuple (c_i, S_{c_i}) using the weak-oracle queries as in Lemma 2.1 and carve out those points and assign them to c_i that satisfy $d_{km}^{est}(c_i, z) \leq r$ for an appropriate value of r. Let c_i belong to the optimal cluster X_i^*. What should the appropriate value of r be such that the points in $S \cap X_i^*$ are guaranteed to get carved out and assigned to c_i? Since the distance estimates are inaccurate, we must use $r = 4r_{opt}$. However, this may cause a point that is $6r_{opt}$ away

Algorithm 2: `Weak-Greedy Ball Carving`

Input : Set of points S, radius Rad
Output: Set $C = \{c_1, \ldots, c_m\}$ and assignment of points in S to centers in C
Initialize: $C = \{\}$
while *S is not empty* **do**
  If ($|C| = k$) **abort**
  If ($|S| \leq 180k \log n$), use strong-oracle queries to find remaining centers covering all points in S within distance $2Rad$. If this is not possible, **abort**. Otherwise, output k centers.
  Pick a subset $T \subset S$ of size $180k \log n$ uniformly at random
  Query the strong-oracle to find distances between elements in T and use these to find a center $c \in T$ such that $|S_c|$ is maximised, where $S_c \equiv |\mathcal{B}(c, 2Rad) \cap T|$. `/* Large sample from cluster */`
  If ($|S_c| < 180 \log n$) **abort**
  Retain $180 \log n$ points in S_c (*i.e., remove the extra point*)
  Assign any point $s \in S$ to center c if $d_{km}^{est}(s, c) \leq 4Rad$ `/* Recover cluster */`
  Add c to C and remove all points assigned to c from S.
end
return *C and the assignments*

from c_i to get carved out and assigned to c_i. This is the reason we obtain an approximation guarantee of 6. This high-level analysis lacks two relevant details: (i) how do we eliminate the assumption that r_{opt} is known, and (ii) probability analysis for all points being assigned a center within distance $6r_{opt}$. We remove the assumption about r_{opt} being known by iterating over discrete choices $Rad = (1+\varepsilon)^0, (1+\varepsilon), (1+\varepsilon)^2, \ldots$ and we argue that the ball carving succeeds for $Rad = r_{opt}(1+\varepsilon)$ with high probability. Hence, the approximation guarantee becomes $6(1+\varepsilon)$. Instead of iterating over the possible choices of Rad linearly, we can do that using binary search, which results in the running time and query complexity getting multiplied by a factor of $O(\log\log\Delta/\varepsilon)$, where Δ is the aspect ratio. Assuming Δ to be polynomially bounded, this factor becomes $O(\log\log n/\varepsilon)$. We describe our k-center algorithm as Algorithm 2 and state the theorem that we prove in Appendix H.

Theorem 3.1. *There exists a $6(1+\epsilon)$-approximation algorithm for the k-center problem that makes $O\left(k^3\log^2 n\log\frac{\log n}{\varepsilon}\right)$ strong-oracle and $O\left(nk\log n\log\frac{\log n}{\varepsilon}\right)$ weak-oracle queries and succeeds with probability at least $(1-1/n^4)^2$.*

4 EXPERIMENTAL RESULTS

We run our algorithms for k-means and k-center problems on synthetic as well as real-world datasets to demonstrate that our algorithms can be implemented efficiently in practice, and provide better results compared to Bateni et al. (2024). Here, we provide experimental results only for k-means on real-world dataset MNIST. The experiments were conducted on a server with 1.5 TB RAM and 64 CPU cores. More details about the experiments including results for k-means on synthetic datasets and experimental results for k-center are given in Appendix I.

Datasets We run our algorithms on synthetic as well as real-world datasets. We generate synthetic data using the Stochastic Block Model (SBM) (Mossel et al., 2015; Abbe, 2018). For the real-world data, we use the MNIST dataset (Deng, 2012; Van der Maaten and Hinton, 2008). The MNIST dataset has ten clusters, $k = 10$. The dataset has $n = 60k$ images, each with a dimension of 28×28. Similar to Bateni et al. (2024), we apply SVD and t-SNE embeddings to embed the dataset into $d = 50$ dimensions and $d = 2$ dimensions, respectively.

Construction of the weak-oracle We create perturbed distance matrices M for the datasets, which are used by the weak-oracles to answer queries. These matrices are initialized with the actual distances between points in the datasets. For the MNIST dataset, if two points belong to the same cluster (i.e., the same digit class), we replace the distance between them with a randomly chosen inter-cluster distance (i.e., the distance between two points from different clusters). If the points belong to different clusters (i.e., different digit classes), we assign them a randomly chosen intra-cluster distance.

Baselines in our experiments Running k-means++ with the strong-oracle distances gives us the *strong-baseline* for our experiments. We calculate the approximation factors by computing the ratio of the cost of the algorithm with this strong baseline.

Experimental results on k-means clustering We run our k-means algorithm on the MNIST dataset using both SVD and t-SNE embeddings. For each embedding, the failure probability δ of the weak-oracle is set to be 0.1, 0.2, and 0.3. For each of these settings, we create a perturbed distance matrix which the weak-oracle uses to answer queries. Then, for each setting, we run our algorithm on the datasets with varying numbers of strong-oracle queries.

Table 2 presents our experimental results for k-means clustering on the MNIST dataset using SVD and t-SNE embeddings. We conducted experiments for different combinations of δ and the number of strong-oracle queries, and we report the values for which the product (number of strong-oracle queries $\times$ log(cost of clustering)) is minimized in Table 2. Table 3 compares the percentage of strong-oracle queries used by our algorithm and by Bateni et al. (2024). Variance figures are reported in Appendix I.

Table 2: Performance of our k-means algorithm on MNIST dataset

MNIST	% of strong-oracle queries			Approximation factor		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$
SVD	0.00013	0.00019	0.00024	1.019	1.030	1.027
t-SNE	0.00054	0.00079	0.00089	1.1595	1.1433	1.0906

Table 3: Comparison of strong-oracle queries for the k-means algorithm on MNIST with Bateni et al. (2024)

MNIST	% of strong-oracle queries (Bateni et al., 2024)			% of strong-oracle queries (Ours)		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$
SVD	0.00062	0.00096	0.00063	0.00013	0.00019	0.00024
t-SNE	0.20969	0.20877	0.43814	0.00054	0.00079	0.00089

Comparison with Bateni et al. (2024): Table 3 shows that our algorithm requires substantially fewer strong-oracle queries compared to Bateni et al. (2024)

for similar approximation guarantees. For MNIST with SVD embedding, the query complexity of our algorithm is at least 61% lower for all values of δ, and for MNIST with t-SNE embedding, the number of queries is at least 99% lower for all δ.

Acknowledgements

We thank the anonymous reviewers for various suggestions to improve the quality of the paper, and in particular to improve the constant factors in the analysis of k-means and k-center algorithms. We have incorporated these suggestions in the appendix. R. Jaiswal acknowledges the support from the SERB, MATRICS grant. R. Jaiswal would like to thank IIT Delhi - Abu Dhabi for hosting him during the later stages of this work. Anup Bhattacharya acknowledges the support from the SERB, CRG grant (CRG/2023/002119).

References

Emmanuel Abbe. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

Raghavendra Addanki, Sainyam Galhotra, and Barna Saha. How to design robust algorithms using noisy comparison oracle. *Proc. VLDB Endow.*, 14(10):1703–1716, 2021. doi: 10.14778/3467861.3467862. URL `http://www.vldb.org/pvldb/vol14/p1703-galhotra.pdf`.

Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 15–28, 2009.

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, 49(4):FOCS17–97, 2019.

Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2018.

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

Zohar Barak, Anupam Gupta, and Inbal Talgam-Cohen. Mac advice for facility location mechanism design. *Advances in Neural Information Processing Systems*, 37:129564–129604, 2024.

MohammadHossein Bateni, Prathamesh Dharangutte, Rajesh Jayaram, and Chen Wang. Metric clustering and mst with strong and weak distance oracles. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 498–550. PMLR, 2024.

Aditya Bhaskara, Sharvaree Vadgama, and Hong Xu. Greedy sampling for approximate clustering in the presence of outliers. *Advances in Neural Information Processing Systems*, 32, 2019.

Anup Bhattacharya, Jan Eube, Heiko Röglin, and Melanie Schmidt. Noisy, greedy and not so greedy k-means++. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, pages 18:1–18:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPICS.ESA.2020.18.

Sanjoy Dasgupta. The hardness of k-means clustering. 2008.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Amit Deshpande and Rameshwar Pratap. Improved outlier robust seeding for k-means. *arXiv preprint arXiv:2309.02710*, 2023.

Lawrence C. Evans. *Partial differential equations*. American Mathematical Society, 2010.

Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

Sainyam Galhotra, Rahul Raychaudhury, and Stavros Sintos. k-clustering with comparison and distance oracles. *Proceedings of the ACM on Management of Data*, 2(5):1–26, 2024.

Buddhima Gamlath, Silvio Lattanzi, Ashkan Norouzi-Fard, and Ola Svensson. Approximate cluster recovery from noisy labels. In Po-Ling Loh and Maxim Raginsky, editors, *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 1463–1509. PMLR, 02–05 Jul 2022. URL `https://proceedings.mlr.press/v178/gamlath22a.html`.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

Christoph Grunau, Ahmet Alper Özüdogru, and Václav Rozhon. Noisy k-means++ revisited. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, pages 55:1–55:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPICS.ESA.2023.55.

Yuzhou Gu and Yinzhan Xu. Optimal bounds for noisy sorting. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1502–1515, 2023.

Richard M Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 881–890. Citeseer, 2007.

Xizhi Liu and Sayan Mukherjee. Tight query complexity bounds for learning graph partitions. In *Conference on Learning Theory*, pages 167–181. PMLR, 2022.

Arya Mazumdar and Barna Saha. Clustering with noisy queries. *Advances in Neural Information Processing Systems*, 30, 2017.

Ramgopal R Mettu and C Greg Plaxton. Optimal time bounds for approximate clustering. *Machine Learning*, 56(1):35–60, 2004.

Adam Meyerson. Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.

Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

Elchanan Mossel, Joe Neeman, and Allan Sly. Consistency thresholds for the planted bisection model. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 69–75, 2015.

Rahul Raychaudhury, Wen-Zhi Li, Syamantak Das, Sainyam Galhotra, and Stavros Sintos. Metric clustering and graph optimization problems using weak comparison oracles. In *Proceedings of Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pages 4777–4830. PMLR, 2025.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. *Advances in neural information processing systems*, 29, 2016.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Improved Algorithms for Clustering with Noisy Distance Oracles: Supplementary Materials

A USEFUL LEMMAS

Lemma A.1. *(Chernoff bounds) (Mitzenmacher and Upfal (2017)) Let $X_1, \ldots, X_n \in \{0,1\}$ be independent random variables for which $\Pr[X_i = 1] = p$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then, the following bounds hold.*

- *For any $0 < \gamma \leq 1$, $\Pr[X \geq (1+\gamma)\mu] \leq e^{-\mu\delta^2/3}$.*
- *For any $0 < \gamma < 1$, $\Pr[X \leq (1-\gamma)\mu] \leq e^{-\mu\delta^2/2}$.*

Fact A.1 (Biased Cauchy–Schwarz Inequality)**.** *(Evans, 2010, Appendix B) For any $\nu > 0$ and real numbers a and b,*

$$(a+b)^2 \leq (1+\nu)(a^2/\nu + b^2).$$

B DETAILED RELATED WORKS

Clustering problems such as k-means and k-center are known to be NP-hard and constant factor approximation algorithms are known for these problems (Dasgupta, 2008; Gonzalez, 1985; Ahmadian et al., 2019). Mettu and Plaxton (2004) prove $\Omega(nk)$ lower bound on the running time of any constant factor randomized approximation of k-means. k-means++ is a D^2-sampling based algorithm that gives an expected $O(\log k)$-approximation for k-means. Aggarwal et al. (2009); Wei (2016) show how to obtain a bi-criteria $(2^{2z}, O(k))$-approximation guarantee for the (k, z)-clustering problem.

Clustering problems have also been studied in semi-supervised settings. Liu and Mukherjee (2022) give tight query complexity bounds for the cluster recovery problem with a membership-query oracle, that on a query with any two points, answers whether the points belong to the same cluster or not. When the goal is to obtain is a good approximate clustering solution, a number of works (Ailon et al., 2018; Mazumdar and Saha, 2017) study clustering problems assuming additional access to a noisy membership-query oracle, also known as the same-cluster query oracle. Gamlath et al. (2022) study algorithms for k-means where noisy labels are provided for each input point using an adversarial or random perturbation of the labels. Bateni et al. (2024) initiate the study of clustering problems in the weak-strong oracle model and design constant approximation algorithms for k-means and k-center problems. They also prove that any constant factor approximation algorithm for k-means or the k-center problem requires $\Omega(k^2)$ strong-oracle queries in the weak-strong model.

Several other noisy oracle models have been studied in the context of clustering. Addanki et al. (2021) introduce two such query models: the adversarial quadruplet oracle and the probabilistic quadruplet oracle. The adversarial quadruplet oracle takes two pairs of points (x_1, y_1) and (x_2, y_2), as input. If the distances between the two pairs are within a constant factor of each other, the oracle returns an arbitrary answer. Otherwise, it correctly identifies which pair has the smaller distance. The probabilistic quadruplet oracle also takes two pairs of points as input. With probability p, it identifies the pair with the smaller distance, and with probability $1-p$, it returns a wrong answer. Addanki et al. (2021) study the k-center clustering problem using access to these noisy oracles along with access to a strong-oracle, and designed a constant factor approximation algorithm. Later, Galhotra et al. (2024) design constant-factor approximation algorithms for both the k-means and k-center problems in the same model, using $\tilde{O}(k^2)$ strong-oracle queries. Recently, Raychaudhury et al. (2025) improve these results by giving constant factor approximation algorithms for both these problems using only $\tilde{O}(1)$ strong-oracle queries.

C DETAILED TECHNICAL OVERVIEW

In this section, we describe the main ideas of our algorithms for k-means and k-center problems in the weak-strong oracle model. One of the most widely used algorithms for k-means is the k-means++ seeding algorithm that works in k iterations. In the first iteration, it chooses a point uniformly at random. Each of the remaining $(k-1)$ iterations samples a point following a non-uniform and adaptive sampling distribution known as the D^2 distribution that depends on the previously chosen centers. Sampling a point from the D^2-distribution is called D^2-sampling. The k points sampled using D^2-sampling are returned as the solution for k-means. The D^2-distribution is such that the probability of choosing a point x is proportional to the squared distance $d(x,C)^2$, where C is a set of chosen centers and $d(x,C) = \min_{c\in C} d(x,c)$. Aggarwal et al. (2009) showed a constant factor bi-criteria approximation for the oversampling version of k-means++ in which $O(k)$ iterations of D^2-sampling are used.

One of the main contributions of this work is to show that one can adapt k-means++ to the weak-strong oracle model. It is easy to observe that one can simulate k-means++ in the strong-oracle model using $O(nk)$ strong-oracle queries. However, things become complicated in the weak-strong model, in which the distances returned by the weak-oracle cannot be trusted. To construct the D^2-sampling distribution, one must calculate $d(x,C)$ for any point x and any set C of centers. Since $d(x,C) = \min_{c\in C} d(x,c)$, if we want to compute this distribution exactly, it would require $\Omega(nk)$ strong-oracle queries. Since we don't want to use too many strong-oracle queries, we proceed as follows. Using weak-oracle query answers, we compute an estimate $d^{est}_{km}(x,C)$ such that with high probability, $d(x,C)$ and $d^{est}_{km}(x,C)$ are within some small additive factor. We construct the sampling distribution to be used by the algorithm using these $d^{est}_{km}(x,C)$ values for all points x.

Let us see how we compute $d^{est}_{km}(x,C)$ for all $x \in X$. The main idea here is to exploit the fact that the weak-oracle distances are wrong independently with probability $\delta < 1/2$. Consider any $c \in C$. Let us obtain an estimate on $d(x,c)$ using only weak-oracle queries. Suppose there are many points in a ball $B(c,r_c)$ around center c of radius r_c for a reasonably small value of r_c. More specifically, suppose r_c is such that $B(c,r_c)$ contains $\Omega(\log n)$ points. Then, the key idea is that the median of the weak-oracle distance queries on pairs (x,y) with $y \in B(c,r_c)$ is a good estimate on $d(x,c)$. More formally, we use $d^{est}_{km}(x,c) = median\{\widetilde{WO}(x,y) : y \in B(c,r_c)\}$ as an estimate for $d(x,c)$. We note that Bateni et al. (2024) also uses the median of query answers to estimate $d(x,c)$. However, their algorithm used these values differently, as we will see in the remaining discussion. Since each weak-oracle query answer is wrong independently with probability at most 1/2, using Chernoff bounds, one can show that for sufficiently large-sized ball $B(c,r_c)$, with high probability, $|d(x,c) - d^{est}_{km}(x,c)| \le r_c$. Now, to approximate $d(x,C)$, we compute $d^{est}_{km}(x,c) + r_c$ for all $c \in C$ and set $d^{est}_{km}(x,C) = \min_{c\in C}\{d^{est}_{km}(x,c) + r_c\}$. Once we have the sampling distribution $\widehat{D^2}$ for which a point x is sampled with probability proportional to $d^{est}_{km}(x,C)^2$, we show that the approach of Aggarwal et al. (2009) can be adapted to obtain a constant factor bi-criteria approximation for k-means that succeeds with high probability. Even though the analysis becomes non-trivial and we incur an approximation loss, the high-level analysis ideas of Aggarwal et al. (2009) goes through.

The main idea is to run the oversampling k-means++ where we iteratively sample and update the center set C. However, the sampling is with respect to $d^{est}_{km}(x,C)^2$ instead of $d(x,C)^2$. The goal is to pick a good center from every optimal cluster. Let us see why oversampling using $d^{est}_{km}(x,C)^2$ will be sufficient to find good centers from every optimal cluster, and hence obtain a good approximation guarantee. For the simplicity of discussion, let us first assume that every optimal cluster has $\Omega(\log n)$ points. We will later see how to drop this assumption. Let us draw a parallel with oversampling k-means++ that samples using accurate distance values. Why does it manage to pick good centers from every optimal cluster? Consider an intermediate center set C. Certain optimal clusters will have a good representative center in C, whereas others remain 'uncovered'. Sampling using the D^2 distribution boosts the probability of sampling from an uncovered cluster, so there is a good chance that the next sample belongs to an uncovered cluster. Moreover, we can argue that given the next sample is from an uncovered cluster, there is a good chance that it will be a good center from that cluster. Let's see whether the same argument applies when sampling uses $d^{est}_{km}(x,C)^2$ instead of $d(x,C)^2$. If our center set C has a single good center (or a few) from an optimal cluster, say A, can we consider the cluster to be covered? No. The probability of sampling from A may remain high since the distance estimate of points in A to the centers in $C \cap A$ may be completely off. When does the distance estimate start becoming tighter? This happens when there are $\Omega(\log n)$ good centers from A in C. This is when we can call the cluster A 'settled'. So, instead of 'covering' every cluster (if correct distances are known), we care about 'settling' every cluster in our current model. So, let us see if we can settle every cluster if we keep sampling centers. We can show that unless the current center set C is already

good, there is a good chance that the next center will be sampled from one of the unsettled optimal clusters. Further, we can also argue that conditioned on sampling from an unsettled cluster, there is a good chance that the sampled center will be a good center (i.e., reasonably close to the optimal center). So, as long as we sample sufficiently many centers, every optimal cluster will get settled with high probability. The oversampling factor is $O(\log n)$, i.e., we end up with a centre set C with $O(k \log n)$ centers. Finally, we will use $d_{km}^{est}(x, C)$ to assign points x to centers and create a weighted point set C (the weight of a point in C is the number of points assigned to it), on which a standard constant approximation algorithm using strong-oracle queries is used to find the final set of k centers. Since the set C has $O(k \log n)$ points, we will need $O(k^2 \log^2 n)$ strong oracle queries to find all the interpoint distances to run the constant approximation algorithm as well as the distance estimates. Using known techniques, it can be shown that this final center set gives a constant approximation. To drop the assumption that all optimal clusters have $\Omega(\log n)$ points, we argue if an optimal cluster does not have adequate points, the oversampling procedure will sample *all* the points from that cluster with high probability, which is also a favourable case.

For the k-center problem, we design a simple ball carving algorithm that gives a $6(1+\varepsilon)$-approximate solution. For the current discussion, we will make the simplifying assumptions that the optimal radius R is known. We will later see how to drop these assumptions. We will design a ball carving procedure that runs in k iterations and is guaranteed to carve out all the points of a new optimal cluster in every round. Carving out a ball means picking a center c and removing all points within a ball of appropriate radius centered at c. As we did for the k-means problem, we'll draw a parallel with the ball carving procedure that works when distances are accurately known. In the perfect case, suppose we pick an arbitrary point as center in each round and carve out the ball of radius $2R$ in every round. For any optimal cluster A and any point $a \in A$, a ball of radius $2R$ will contain all points in A. So, in every round, the center c is guaranteed to belong to a previous uncarved ball and we are guaranteed to carve out all points in the optimal cluster to which c belongs. So, in k iterations, we will carve out all points within radius $2R$. This gives us a 2-approximate solution. Let's see if the same argument applies to our current weak-strong oracle model. The main issue while carving a ball centered at the next chosen center c is that we may not have accurate distance estimates. What should be the appropriate center c and the appropriate radius r so that if we carve out all points whose distance estimate to c is within r, then all points from the optimal cluster to which c belongs get removed? Suppose all points from $i-1$ optimal clusters have been carved out in the first $i-1$ iterations. Let's focus on the ith iteration. If only a few points are remaining, then we can find the centers using strong queries for all remaining points. Let us assume that at least $\Omega(k \log n)$ points remain. We can argue that if we randomly sample a set T of $\Omega(k \log n)$ points from the remaining set of points, then there is a point $c \in T$ such that there are $\Omega(\log n)$ points in $B(c, 2R) \cap T$. We can use strong-oracle queries on the set T to locate center c and $\Omega(\log n)$ points from T within $2R$ of c. Let c belong to the optimal cluster A. So, if we consider the closest $O(\log n)$ points to c when calculating the median distance (this is the same as we did for k-means), then our distance estimate is off by an additive factor of $2R$. This means that if we carve out all points within the estimated distance $4R$ from c, it is guaranteed to remove all points in A. However, the removed points may also include points with true distance $6R$ away from c, which gets assigned to c. So, we will end with an approximation guarantee of 6. What is the number of strong oracle queries required for the procedure? In each round, we need $O(k^2 \log^2 n)$ strong queries and there are k rounds. So, the total number of strong queries is $O(k^3 \log^2 n)$. How do we remove the assumption that the optimal radius R is known? This is done by guessing the radius using binary search in an appropriate range and out of a discrete set of possibilities determined by the error parameter $\varepsilon > 0$. This makes the approximation guarantee $6(1+\varepsilon)$ and adds a multiplicative factor of $\log \frac{\log n}{\varepsilon}$ to the number of strong oracle queries.

Bateni et al. (2024) show a $\Omega(k^2)$ strong-oracle query lower bound in the weak-strong model for any constant factor approximation of k-means or k-center problems. However, no such weak-oracle query lower bound was known for k-means. In this work, we show a $\Omega(\frac{nk}{(1-2\delta)^2})$ weak-oracle query lower bound for any constant approximation of k-means in the weak-oracle model. To show this lower bound, we use a known $\Omega(\frac{nk}{(1-2\delta)^2})$ query lower bound of Mazumdar and Saha (2017) for the cluster recovery problem using a faulty query oracle.

D INFORMAL JUSTIFICATION OF THE WEAK-ORACLE MODEL:

In the weak-oracle model, it is assumed that the weak-oracle returns the exact distance between any two points independently with probability $(1-\delta)$ and with the remaining probability, it can report any arbitrary value. We first note that one can relax the requirement that the weak-oracle needs to report the exact distance between

two points as follows. The weak-oracle could return a value within a constant factor of the actual distance, with high probability. Our algorithms would work with this modified weak-oracle and will give a solution of quality at most a constant factor worse compared to the earlier solution. Regarding the independence assumption of the weak-oracle, we note that our algorithms should also work in bounded independence models. We apply Chernoff-Hoeffding under independence, which can be replaced with other concentration inequalities under bounded independence. However, the algorithms do not work if the query answers are arbitrarily correlated. We are not aware of many studies that design algorithms for clustering problems that work with arbitrarily correlated Oracle answers. One recent work along these lines is Barak et al. (2024), which designs mechanisms for facility location problems given noisy Oracle access (oracle gives noisy locations for the clients) and highlights potential difficulties in obtaining strong algorithmic results without the independence assumption. On the other hand, there are applications where assuming that the errors are independent is not completely unreasonable, such as when distances are obtained using a crowd-sourced platform. So, the independence assumption provides a good initial platform for understanding important questions, such as whether it is possible to retain the approximation guarantees of widely used algorithms such as k-means++ (for k-means) and ball-carving (for k-center) in this model.

One natural attempt to model noisy distances might be where the weak-oracle answers are values sampled from a probability distribution. First, let us consider the scenario in which the oracle answers are not persistent. In this case, one can ask the same query multiple times and obtain concentration bounds for the queried distances. This makes the problem easy. Now, let us assume that the query answers are persistent. If the probability distributions have strong concentration properties, then with high probability (say, with probability $(1-\delta)$), the samples will be close to the expected value and with remaining probability (with probability at most δ), the sample will be far from the expected value. We believe that this can be modeled in the weak-strong oracle model that we consider in this paper with an appropriate choice of parameters. First, note that one can relax the weak-oracle to require that it only returns an answer within a constant factor of the actual value, and our algorithms with this modified oracle will return solutions that is at most a constant factor worse. The above discussion assumes that the distances are drawn from independent probability distributions and the sampling is also done independently.

E CLUSTERING IN STRONG-ORACLE MODEL

Strong-oracle model: We note that strong-oracle model is precisely the well-known classical model. So, let us remind ourselves of what is known. We observe that k-means++ algorithm can be executed using $O(nk)$ strong-oracle queries and using known results, one can show that it gives an $O(\log k)$-approximation in expectation for the k-means problem (Arthur and Vassilvitskii, 2007). Further, the oversampling version of k-means++ gives constant factor bi-criteria approximation for k-means using only $O(nk)$ strong-oracle queries (Aggarwal et al., 2009). We also note that a $\Omega(nk)$ strong-oracle query lower bound for any constant factor approximation of k-means follows from Mettu and Plaxton (2004). Even though we state results only for the k-means problem, it is possible to extend these ideas to obtain a $O(2^{2z})$ bi-criteria approximation in expectation for the (k,z)-clustering using $O(nk)$ strong-oracle queries (Wei, 2016). For the k-center problem, we observe that $O(nk)$ strong-oracle queries suffice to run Gonzalez's farthest-point algorithm to obtain a 2-approximation (Gonzalez, 1985). These results are discussed in Appendix E.

E.1 Algorithm for k-means Clustering

Theorem E.1. *There exists an algorithm that makes $O(nk)$ strong-oracle queries and returns a constant factor bi-criteria approximation for k-means with constant probability.*

Proof. We show that the oversampling k-means++ algorithm, which samples centers using D^2-sampling for $O(k)$ iterations, can be simulated using $O(nk)$ strong-oracle queries. Recall that oversampling k-means++ starts with a center c_1 chosen uniformly at random and initializes a center set $C = c_1$. We query the strong-oracle with (x, c_1) for all $x \in X$ to obtain $SO(x, c_1)$ for all $x \in X$, and construct the D^2-sampling distribution. This step requires $O(n)$ strong-oracle queries. In the second iteration, the oversampling k-means++ algorithm samples a center c_2 following D^2-distribution with respect to $C = \{c_1\}$. We update $C = C \cup \{c_2\}$. We again query the strong-oracle for $x \in X$ and c_2, and update the sampling distribution using the query results. In this manner for

$O(k)$ iterations, we make in total $O(nk)$ strong-oracle queries and using the query results, run the oversampling k-means++ algorithm. Following Aggarwal et al. (2009), oversampling k-means++ gives a constant approximation for k-means with constant probability. □

The above bound on the number of strong-oracle queries is tight for a constant factor approximation of k-means, following a known lower bound Mettu and Plaxton (2004).

Theorem E.2. *Any randomized algorithm that provides a constant-factor approximation for the k-means problem with constant probability requires $\Omega(nk)$ strong-oracle queries.*

Proof. Mettu and Plaxton (Mettu and Plaxton, 2004) prove a $\Omega(nk)$ lower bound on the running time of any constant factor randomized approximation of k-means. The $\Omega(nk)$ query lower bound result follows from this lower bound of Mettu and Plaxton (2004). □

E.2 Algorithm for k-center Clustering

Theorem E.3. *There exists a 2-approximation algorithm for the k-center problem using $O(nk)$ queries.*

Proof. We observe that the farthest-point algorithm of Gonzalez (1985) can be implemented using only $O(nk)$ strong-oracle queries. This will give a 2-approximation for the k-center problem. □

F ALGORITHM FOR K-MEANS IN WEAK-STRONG ORACLE MODEL

For ease of reading, we reproduce here the entire algorithm for k-means clustering and its analysis. In this section, we design a constant factor bi-criteria approximate solution for k-means in the weak-strong oracle model with the assumption that all optimal clusters have size at least $\frac{480 \log n}{\epsilon}$. We remove this assumption later. We prove the following result.

Theorem F.1. *Let $\epsilon \in (0,1)$, $\delta \leq 1/3$. There exists a randomized algorithm for k-means that makes $O(\frac{k^2 \log^2 n}{\epsilon^6})$ strong-oracle queries and $O(\frac{nk \log n}{\epsilon^3})$ weak-oracle queries to give a $\left(O(\frac{\log n}{\epsilon^3}), 40(1+\epsilon)\right)$ bi-criteria approximation for k-means and succeeds with constant probability, assuming every optimal cluster has size at least $\frac{480 \log n}{\epsilon}$.*

F.1 Algorithm and Analysis

Since the weak-oracle gives a wrong answer to a query independently with probability $\delta < 1/2$, we define an alternate distance measure between points and a set of centers that we use in our algorithm. Let C denote a set of centers. For any $x \in X$ and any $c \in C$, we query for the distance between x and c to the weak oracle $\widetilde{WO}$ to obtain a possibly wrong answer $\widetilde{WO}(x,c)$. For any fixed x, we query the weak-oracle for each center c in C, and use these query answers $\widetilde{WO}(x,c)$ to come up with an upper bound on the distance between a point x and a set C of centers defined as follows. We use $\delta \leq 1/3$ for this discussion.

Definition F.1. *Let x and y be any two points in X and we want to estimate the distance between x and y. Let radius $r_y \geq 0$ be such that the ball $B(y, r_y)$ contains at least $180 \log n$ points. Then, for any $x \in X$, we define the distance from x to y as $d^{est}_{km}(x,y) = median\{\widetilde{WO}(x,z) | z \in B(y, r_y)\}$.*

We make the following claim with respect to the above distance measure. A similar lemma was proved in Bateni et al. (2024). We add a proof here for completeness.

Lemma F.1. *Following Definition F.1, for any $x \in X$, with probability at least $\left(1 - \frac{1}{n^5}\right)$, we have $|d^{est}_{km}(x,y) - d(x,y)| \leq r_y$.*

Proof. Let us consider the ball $B(y, r_y)$ centered at y and radius $r_y \geq 0$. By definition, $B(y, r_y)$ has at least $l = 180 \log n$ points inside it. For each point $z \in B(y, r_y)$, the weak-oracle returns a wrong answer for $\widetilde{WO}(x,z)$ with probability $\delta \leq 1/3$. Hence, the expected number of wrong answers is at most $l/3$. For each query to the weak-oracle with x and $z \in B(y, r_y)$, we have $|\widetilde{WO}(x,z) - d(x,z)| > r_y$ with probability at most $1/3$. For the median of the query answers $d^{est}_{km}(x,y) = median\{\widetilde{WO}(x,z) : z \in B(y, r_y)\}$ to satisfy $|d^{est}_{km}(x,y) - d(x,y)| > r_y$,

we need at least half of the query answers $\widetilde{WO}(x,z)$ to satisfy $|\widetilde{WO}(x,z) - d(x,z)| > r_y$. Using Lemma A.1, this probability is at most $e^{-1/3 \cdot 1/4 \cdot l/3} \leq 1/n^5$. □

We use the above definition to come up with an upper bound on the distance between a point x and a set C of centers. For any $c \in C$, let r_c denote a radius such that $B(c, r_c)$ has at least $180 \log n$ points inside it. We use Definition F.1 to come up with an upper bound on the distance between x and any $c \in C$ as $d^{est}_{km}(x,c) + r_c$. Following Lemma F.1, with high probability, this upper bound holds. That is, with high probability, $d(x,c) \leq d^{est}_{km}(x,c) + r_c$. Using an union bound over all $c \in C$, all these upper bounds hold and hence, the minimum of these upper bounds would also hold. This motivates the following definition on the distance between a point x and a set C of centers.

Definition F.2. *Let x be any point in X and let C be a set of centers of size at least $180 \log n$. For any $c \in C$, let r_c denote a radius value such that the ball $B(c, r_c)$ contains at least $180 \log n$ points. We define the distance between x and C as $d^{est}_{km}(x,C) = \min_{c \in C}\{d^{est}_{km}(x,c) + r_c\}$.*

We state the following lemma with respect to the above distance measure.

Lemma F.2. *Let x be any point in X and let C denote a set of centers of size at least $180 \log n$. Then, with probability at least $(1 - 1/n^4)$, $d(x,C) \leq d^{est}_{km}(x,C)$. In other words, with probability at least $(1 - 1/n^4)$, there exists a center $c \in C$ within a distance of $d^{est}_{km}(x,C)$ from x.*

Proof. Let x be any point in X. Using Lemma F.1, for any $c \in C$, we have with probability at least $(1 - 1/n^5)$, $d^{est}_{km}(x,c) + r_c \geq d(x,c)$. Using a union bound over all $c \in C$, with probability at least $(1 - 1/n^4)$, all these upper bounds on $d(x,c)$ hold. Hence, $d^{est}_{km}(x,C) = \min_{c \in C}\{d^{est}_{km}(x,c) + r_c\} \geq \min_{c \in C} d(x,c) = d(x,C)$, with probability at least $(1 - 1/n^4)$. Since with probability at least $(1 - 1/n^4)$, these upper bounds $d(x,c) \leq d^{est}_{km}(x,c) + r_c$ hold for $c \in C$, and the minimum over these upper bounds also hold. This implies that with this probability, there exists a center $c \in C$ within distance $d^{est}_{km}(x,C)$ from x. □

Estimating $d^{est}_{km}(x,C)$ in weak-strong oracle model Next, we describe how we estimate $d^{est}_{km}(x,C)$ for all $x \in X$ in the weak-strong oracle model. We assume that there are at least $180 \log n$ centers in C. Consider any $c \in C$. We query the strong-oracle SO with all pairs $c_i, c_j \in C$ to obtain the exact distances between centers c_i and c_j as $SO(c_i, c_j)$. For each $c \in C$, we find the smallest r_c such that $B(c, r_c)$ contains at least $180 \log n$ points in C. Consider any $x \in X$ for which we want to estimate $d^{est}_{km}(x,C)$. We query the weak-oracle with x and $y \in B(c, r_c)$ to obtain $\widetilde{WO}(x,y)$ for all $y \in B(c, r_c)$, and use these weak-oracle query answers to compute $d^{est}_{km}(x,c)$ following Definition F.1. We compute $d^{est}_{km}(x,c)$ for all $c \in C$. Finally, to compute $d^{est}_{km}(x,C)$, we find the minimum over all $c \in C$ of $d^{est}_{km}(x,c) + r_c$, as mentioned in Definition F.2.

Let C be any set of centers of size at least $180 \log n$. Let $\widetilde{D^2}$ denote the proportional sampling distribution over points in X with respect to C. The probability of sampling a point $x \in X$ following distribution $\widetilde{D^2}$ is proportional to $d^{est}_{km}(x,C)^2$. For any point $x \in X$, the probability of sampling x with respect to C is given as $\frac{d^{est}_{km}(x,C)^2}{\sum_{x' \in X} d^{est}_{km}(x',C)^2}$. Using Lemma F.2 with a union bound over all points $x \in X$, we have that with probability at least $(1 - \frac{1}{n^3})$, the distance estimates $d^{est}_{km}(x,C)$ given by Definition F.2 provide valid upper bounds for $d(x,C)$ for all $x \in X$. For the following discussion, we condition on this high probability event $\mathcal{E}$.

Lemma F.3. *Let C be any set of centers of size at least $180 \log n$. Let $\mathcal{E}$ denote the event that $d^{est}_{km}(x,C) \geq d(x,C)$ for all points $x \in X$. Then, $\mathcal{E}$ holds with probability at least $1 - 1/n^3$.*

Proof. The proof follows by using Lemma F.2 for all $x \in X$ and applying a union bound. □

Since we do not know the exact distances for most pairs of points in a cluster, we define below the cost of a cluster using the distance estimates defined in Definition F.2.

Definition F.3. *(Cost of a cluster) Let A be any optimal cluster and let C denote a set of centers. Then, the cost of cluster A with respect to center set C is given as $\tilde{\Phi}(A,C) = \sum_{x \in A} d^{est}_{km}(x,C)^2$.*

We next define the notion of *settled* clusters that give good approximation guarantees with respect to the set C of centers chosen by the algorithm.

Algorithm 3: Algorithm for k-means in weak-strong oracle model

Input : Dataset X and an integer $k > 0$, $\epsilon \in (0,1)$, $\delta = 1/3$.
Output: A set of $O(\frac{k \log n}{\epsilon^3})$ centers and an assignment of $x \in X$ to centers.
▷ Let C_1 be a set of $180 \log n$ points chosen arbitrarily from X.;
Set $t = \frac{4320 \cdot 29160}{\epsilon^3} \cdot k \log n$;
for $i = 1$ **to** t **do**
  ▷ For each $x \in X$, compute $d^{est}_{km}(x, C_i)$ following Definition F.2.;
  ▷ Construct distribution $\widetilde{D^2}$ that samples a point $x \in X$ with probability proportional to $d^{est}_{km}(x, C_i)^2$.;
  ▷ Sample a point $s_i \in X$ using distribution $\widetilde{D^2}$.;
  ▷ Make a strong-oracle query with $SO(s_i, c)$ for all $c \in C_i$.;
  ▷ Update $C_{i+1} \leftarrow C_i \cup \{s_i\}$.
end
▷ Let $\{c_1, c_2, \ldots, c_h\}$ be an arbitrary ordering of the centers in C_{t+1}, where $h = 180 \log n + t$.
▷ Initialize weights $w(c_i) = 0$ for $i \in [h]$.
for $x \in X$ **do**
  ▷ Compute $d^{est}_{km}(x, C_{t+1})$ and determine c_x for which the minimum is achieved in Definition F.2.;
  ▷ Assign x to c_x.;
  ▷ Update $w(c_x) \leftarrow w(c_x) + 1$.
end
return C_{t+1} *and the assignment of each* $x \in X$ *to a center in* C_{t+1}.

Definition F.4. *(Settled clusters) Let C be a set of centers. An optimal cluster A is said to be settled with respect to C if $\tilde{\Phi}(A, C) \leq 20(1+\epsilon) \cdot OPT(A)$. Otherwise, the cluster A is said to be unsettled.*

Let C_{i-1} denote the set of centers at the end of iteration $(i-1)$. Let A be an optimal cluster. Let the set of *settled* optimal clusters at the start of iteration i be given as $G_i = \{A : \tilde{\Phi}(A, C_{i-1}) \leq 20(1+\epsilon) \cdot OPT(A)\}$ and the set of *unsettled* optimal clusters be given as $B_i = \{A : \tilde{\Phi}(A, C_{i-1}) > 20(1+\epsilon) \cdot OPT(A)\}$.

Lemma F.4. *During the ith iteration of Algorithm 3, either $\tilde{\Phi}(X, C_{i-1}) \leq 40(1+\epsilon) \cdot OPT$ or the probability of sampling a center from some unsettled optimal cluster $A \in B_i$ using distribution $\widetilde{D^2}$ is at least 1/2.*

Proof. If $\tilde{\Phi}(X, C_{i-1}) \leq 40(1+\epsilon) \cdot OPT$, then C_{i-1} gives a $40(1+\epsilon)$-approximation. For the following, we assume $\tilde{\Phi}(X, C_{i-1}) > 40(1+\epsilon) \cdot OPT$. We show the probability of sampling a point from an unsettled optimal cluster $A \in B_i$ is at least 1/2. Let x be a point sampled using distribution $\widetilde{D^2}$. Then,

$$\begin{aligned}
\Pr(x \in A, A \in B_i) &= \frac{\sum_{A \in B_i} \tilde{\Phi}(A, C_{i-1})}{\tilde{\Phi}(X, C_{i-1})} \\
&= 1 - \frac{\sum_{A \in G_i} \tilde{\Phi}(A, C_{i-1})}{\tilde{\Phi}(X, C_{i-1})} \\
&\geq 1 - 20(1+\epsilon) \cdot \frac{\sum_{A \in G_i} OPT(A)}{40(1+\epsilon) \cdot OPT} \\
&\geq 1 - 1/2 = 1/2
\end{aligned}$$

This completes the proof of the lemma. □

Let $A \in B_i$ be an unsettled optimal cluster. Let μ_A be the centroid of A. We define r to be the average radius of cluster A given as $r = \sqrt{OPT(A)/|A|}$. We define an inner ring $IR(A, \alpha)$ around μ_A of radius αr, for some $\alpha > 1$ to be chosen later. Formally, $IR(A, \alpha) = \{x \in A : ||x - \mu_A|| \leq \alpha r\}$.

One of the main ideas in the analysis of Algorithm 3 is to show that once we sample at least $180 \log |A|$ centers from the inner ring of an unsettled optimal cluster A, the cost of cluster A with respect to the set of centers becomes at most a constant times its optimal cost $OPT(A)$ with high probability. Fix any point $x \in A$. Let C_i denote the set of centers at the end of iteration i such that it contains at least $180 \log |A|$ points from the inner

ring $IR(A,\alpha)$ of the optimal cluster A. Using arguments similar to Lemma F.3, we can show with probability at least $(1-1/|A|^3)$, there exists a center c_x in C_i within distance $d^{est}_{km}(x,C_i)$ from x and this holds for all $x \in A$. Since C_i has at least $180\log|A|$ points from the inner ring of A, with probability at least $(1-1/|A|^3)$, the distance $d^{est}_{km}(x,C_i)$ from x to c_x is at most the distance from x to its farthest point in the inner ring of A plus the diameter of the inner ring. This corresponds to $c_x \in C_i \cap IR(A,\alpha)$ being the farthest point in the inner ring from x and $r_{c_x} \leq 2\alpha r$.

Lemma F.5. *Let C_{i-1} denote the set of sampled centers at the end of iteration $(i-1)$, and let us assume that during iteration i, we have sampled a point such that C_i has at least $180\log n$ points from the inner ring $IR(A,\alpha)$ of A, where A is an unsettled optimal cluster with respect to C_{i-1}. For $\alpha > 1$, we have $\tilde{\Phi}(A,C_i) \leq 2(1+9\alpha^2)\cdot OPT(A)$ with probability at least $(1-1/|A|^3)$.*

Proof. Following Lemma F.3, with probability at least $(1-1/n^3)$, for all points $x \in A$, we have $d^{est}_{km}(x,C_i)$ to be at most $d^{est}_{km}(x,c_x)+r_{c_x}$, where $c_x \in C_i \cap IR(A,\alpha)$ with radius $r_{c_x} \leq 2\alpha r$. We have the following.

$$\begin{aligned}
\tilde{\Phi}(A,C_i) &= \sum_{x\in A} d^{est}_{km}(x,C_i)^2 \\
&\leq \sum_{x\in A} (d(x,c_x)+2\alpha r)^2, \text{ where } c_x \in C_i \cap IR(A,\alpha) \\
&\leq \sum_{x\in A} (d(x,\mu_A)+3\alpha r)^2 \qquad (1)\\
&\leq \sum_{x\in A} [2d^2(x,\mu_A)+18\alpha^2 r^2] \\
&\leq 2OPT(A) + 18\sum_{x\in A}(\alpha r)^2 \\
&\leq 2OPT(A) + 18\cdot|A|\cdot\alpha^2\frac{OPT(A)}{|A|} \\
&= 2(1+9\alpha^2)\cdot OPT(A)
\end{aligned}$$

This completes the proof of the lemma. □

Next, we show that the probability of sampling a point following distribution $\widetilde{D^2}$ from the inner ring $IR(A,\alpha)$ of an unsettled optimal cluster A is not too small. First, we show that the inner ring of an unsettled optimal cluster A contains a significant fraction of points of A.

Lemma F.6. *For any unsettled optimal cluster A, there are at least $|A|(1-\frac{1}{\alpha^2})$ points in the inner ring $IR(A,\alpha)$ of that cluster, where $\alpha > 1$.*

Proof. We have:

$$\begin{aligned}
OPT(A) &\geq \sum_{x\in A\setminus IR(A,\alpha)} ||x-\mu_A||^2 \\
&\geq |A\setminus IR(A,\alpha)|(\alpha r)^2 \\
&= \left(1-\frac{|IR(A,\alpha)|}{|A|}\right)|A|(\alpha r)^2 \\
&= \left(1-\frac{|IR(A,\alpha)|}{|A|}\right)\alpha^2 OPT(A)
\end{aligned}$$

Rearranging terms, we get $|IR(A,\alpha)| \geq |A|(1-\frac{1}{\alpha^2})$. □

Lemma F.6 tells us that the inner ring of an unsettled optimal cluster A contains at least $(1-1/\alpha^2)$-fraction points of cluster A. Conditioned on sampling a point from an unsettled optimal cluster A following distribution $\widetilde{D^2}$ during the ith iteration, the probability of sampling a point from the inner ring of A is $\frac{\tilde{\Phi}(IR(A,\alpha),C_{i-1})}{\tilde{\Phi}(A,C_{i-1})}$. To get a lower bound on this probability, we need a lower bound on $\tilde{\Phi}(IR(A,\alpha),C_{i-1})$ and an upper bound on $\tilde{\Phi}(A,C_{i-1})$. We show the following.

Lemma F.7. $\Pr(x \in IR(A,\alpha) \mid x \in A, A \in B_i) \geq (1-1/\alpha^2)\frac{(\sqrt{1+10\epsilon/9}-\alpha)^2}{180(1+\epsilon)}$.

Proof. Conditioned on sampling a point from an unsettled optimal cluster A, the probability of sampling a point from the inner ring of that cluster during the ith iteration is given as

$$\Pr(x \in IR(A,\alpha) \mid x \in A, A \in B_i) = \frac{\tilde{\Phi}(IR(A,\alpha), C_{i-1})}{\tilde{\Phi}(A, C_{i-1})}$$

Let c_{μ_A} denote the center in C_{i-1} for which $d^{est}_{km}(\mu_A, C_{i-1})$ is minimized. Let $v = d^{est}_{km}(\mu_A, C_{i-1})$. Recall that for any point $x \in A$, c_x denotes the center in C_{i-1} for which $d^{est}_{km}(x, C_{i-1})$ is minimized.

Claim F.1. *Let x and y be any two points for which $d^{est}_{km}(x, C_{i-1})$ and $d^{est}_{km}(y, C_{i-1})$ are minimized by centers $c_x \in C_{i-1}$ and $c_y \in C_{i-1}$, respectively. Then, assuming event $\mathcal{E}$ holds for all iterations of Algorithm 3, we have $d^{est}_{km}(y, c_x) \leq d(x,y) + d^{est}_{km}(x, c_x) + 2r_{c_x}$ and $d^{est}_{km}(x, c_y) \leq d(x,y) + d^{est}_{km}(y, c_y) + 2r_{c_y}$.*

Proof. Let x be a point in the optimal cluster A. Since $d^{est}_{km}(x, C_{i-1})$ is minimized by $c_x \in C_{i-1}$, for point x, the center c_x minimizes $\min_{c \in C_{i-1}}(d^{est}_{km}(x,c) + r_c)$ and the ball $B(c_x, r_{c_x})$ has at least $180 \log n$ points in it. Conditioning on event $\mathcal{E}$ for all iterations of Algorithm 3, we have for both $d^{est}_{km}(x, c_x)$ and $d^{est}_{km}(y, c_x)$, the median distances are realized by points inside the ball $B(c_x, r_{c_x})$. Hence, $d^{est}_{km}(y, c_x) \leq d(x,y) + d^{est}_{km}(x, c_x) + 2r_{c_x}$. Similarly, the other inequality holds. □

Next, we obtain an upper bound on $\tilde{\Phi}(A, C_{i-1})$.

$$\begin{aligned}
\tilde{\Phi}(A, C_{i-1}) &= \sum_{x \in A} d^{est}_{km}(x, C_{i-1})^2 \\
&= \sum_{x \in A} (d^{est}_{km}(x, c_x) + r_{c_x})^2 \\
&\leq \sum_{x \in A} (d^{est}_{km}(x, c_{\mu_A}) + r_{c_{\mu_A}})^2 \\
&\leq \sum_{x \in A} (d(x, \mu_A) + d^{est}_{km}(\mu_A, c_{\mu_A}) + 3r_{c_{\mu_A}})^2 \quad \text{(using Claim F.1)} \\
&\leq \sum_{x \in A} (d(x, \mu_A) + 3d^{est}_{km}(\mu_A, C_{i-1}))^2 \qquad (2) \\
&\leq \sum_{x \in A} (2d(x, \mu_A)^2 + 18 d^{est}_{km}(\mu_A, C_{i-1})^2) \\
&= \sum_{x \in A} (2d(x, \mu_A)^2 + 18v^2) \\
&= 2OPT(A) + 18|A|v^2 \\
&= 2|A|(r^2 + 9v^2)
\end{aligned}$$

Now, we compute a lower bound on $\tilde{\Phi}(IR(A,\alpha), C_{i-1})$. Let b be any point in the inner ring $IR(A,\alpha)$ of optimal cluster A and let c_b be the center in C_{i-1} for which $d^{est}_{km}(b, C_{i-1})$ is minimized. We observe that $d^{est}_{km}(\mu_A, C_{i-1}) \leq d^{est}_{km}(\mu_A, c_b) + r_{c_b}$. Using Claim F.1, $d^{est}_{km}(\mu_A, c_b) + r_{c_b} \leq d(\mu_A, b) + d^{est}_{km}(b, c_b) + 3r_{c_b} \leq d(\mu_A, b) + 3d^{est}_{km}(b, C_{i-1})$. Hence, $d^{est}_{km}(b, C_{i-1}) \geq (d^{est}_{km}(\mu_A, C_{i-1}) - d(\mu_A, b))/3 = (v - \alpha r)/3$. We obtain the following.

$$\begin{aligned}
\tilde{\Phi}(IR(A,\alpha), C_{i-1}) &= \sum_{b \in IR(A,\alpha)} d^{est}_{km}(b, C_{i-1})^2 \\
&\geq |IR(A,\alpha)| \frac{(v-\alpha r)^2}{9} \\
&\geq |A| \left(1 - \frac{1}{\alpha^2}\right) \frac{(v-\alpha r)^2}{9}
\end{aligned}$$

Therefore, we have $\Pr(x \in IR(A,\alpha) \mid x \in A, A \in B_i) \geq \frac{(1-1/\alpha^2)(v-\alpha r)^2}{18(r^2+9v^2)}$. Since A is an unsettled optimal cluster with respect to C_{i-1}, we cannot have $v = d^{est}_{km}(\mu_A, C_{i-1})$ to be very small. We obtain a lower bound on v as follows. Since A is an unsettled optimal cluster with respect to C_{i-1}, we have $\tilde{\Phi}(A, C_{i-1}) > 20(1+\epsilon) \cdot OPT(A)$.

$$\begin{aligned} 20(1+\epsilon) \cdot OPT(A) &< \tilde{\Phi}(A, C_{i-1}) \\ &\leq 2(OPT(A) + 9|A|v^2)) \end{aligned} \tag{3}$$

Simplifying, we get

$$v > \sqrt{\frac{(18+20\epsilon) \cdot OPT(A)}{18|A|}} = \sqrt{(1+10\epsilon/9)} \cdot r \tag{4}$$

Substituting the above value for v, we get the probability of sampling a point from the inner ring of an unsettled optimal cluster A, conditioned on sampling a point from A is at least $\Pr(x \in IR(A,\alpha) \mid x \in A, A \in B_i) \geq (1-1/\alpha^2)\frac{(\sqrt{1+10\epsilon/9}-\alpha)^2}{180(1+\epsilon)}$. □

For $\alpha = 1+\epsilon/3$, we have $2(1+9\alpha^2) \leq 20(1+\epsilon)$. For this choice of α, following Lemma F.5, we obtain that the cost of an optimal cluster becomes at most $20(1+\epsilon)$ times its optimal cost once $180 \log n$ points are sampled from its inner ring. The probability of sampling a point from the inner ring of A becomes $\Pr(x \in IR(A,\alpha) \mid x \in A, A \in B_i) \geq (1 - \frac{1}{(1+\epsilon/3)^2})\frac{(\sqrt{1+10\epsilon/9}-(1+\epsilon/3))^2}{180(1+\epsilon)} \geq \frac{\epsilon^3}{29160} = g(\epsilon)$, where $g(\epsilon) = \frac{\epsilon^3}{29160}$. We show that during iteration i of Algorithm 3, a point sampled following the $\widetilde{D^2}$ distribution will belong to the inner ring of an unsettled optimal cluster with probability at least $g(\epsilon)/2$. We bound the number of required samples such that the sampled set C contains at least $180 \log n$ points from the inner rings of all unsettled optimal clusters with high probability.

Lemma F.8. *Let $C \subset X$ denote a set of $\frac{4320k \log n}{g(\epsilon)}$ points sampled using distribution $\widetilde{D^2}$ in Algorithm 3. Then, the total number of samples from the inner rings of unsettled optimal clusters in C is at least $180k \log n$ with probability at least $1 - \frac{1}{n^k}$.*

Proof. Let C denote a set of $\frac{4320k \log n}{g(\epsilon)}$ points sampled following distribution $\widetilde{D^2}$ in Algorithm 3. We will show that C contains at least $180k \log n$ points from the inner rings of unsettled optimal clusters with high probability. Let s_i be a point sampled during iteration i. We define a random variable X_i as follows:

$$X_i = \begin{cases} 1 & \text{if } s_i \text{ belongs to the inner ring of an unsettled optimal cluster} \\ 0 & \text{otherwise} \end{cases}$$

Let $l = \frac{4320k \log n}{g(\epsilon)}$. Then, $X = \sum_{i=1}^{l} X_i$ represents the total number of sampled points that belong to the inner rings of unsettled optimal clusters. From Lemma F.4 and Lemma F.7, the probability that a point sampled using distribution $\widetilde{D^2}$ belongs to the inner ring of an unsettled optimal cluster is:

$$\begin{aligned} \Pr[X_i = 1] &= \Pr[x \in IR(A,\alpha) : A \in B_i] \\ &= \Pr[x \in IR(A,\alpha) \mid x \in A, A \in B_i] \cdot \Pr[x \in A, A \in B_i] \\ &\geq \frac{1}{2} \cdot g(\epsilon) \end{aligned}$$

Therefore, we have $\mathbb{E}[X_i] \geq \frac{1}{2} \cdot g(\epsilon)$. The expected number of centers in C that belong to the inner rings of unsettled optimal clusters is $\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{l} X_i\right] = \sum_{i=1}^{l} \mathbb{E}[X_i] \geq l \cdot \frac{1}{2} \cdot g(\epsilon) = 2160k \log n$. Using Chernoff bounds (Lemma A.1), the probability $\Pr[X < 180k \log n] = \Pr[X < \left(1 - \frac{11}{12}\right) 2160k \log n] \leq e^{-2160k \log n \cdot \frac{121}{144} \cdot \frac{1}{2}} \leq \frac{1}{n^k}$. □

Lemma F.9. *With probability at least $(1-1/n^k)(1-1/n^2)$, all optimal clusters become settled once a set C of $\frac{4320k \log n}{g(\epsilon)}$ points are sampled following distribution $\widetilde{D^2}$ in Algorithm 3.*

Proof. Following Lemma F.8, with probability at least $(1-1/n^k)$, a set C of $\frac{4320k\log n}{g(\epsilon)}$ points sampled following distribution $\widetilde{D^2}$ contains at least $180k\log n$ points from the inner rings of unsettled optimal clusters. Using Lemma F.5, we know that once $180\log n$ points are sampled from the inner ring of an unsettled optimal cluster A, with probability at least $1-1/n^3$, that cluster becomes settled. We condition on the event that unsettled optimal clusters become settled once $180\log n$ points are sampled from their inner rings, and using a union bound over k optimal clusters, this event happens for all k optimal clusters with probability at least $(1-1/n^2)$.

Note that as long as there is an unsettled optimal cluster, a sampled point in C will belong to the inner ring of some unsettled optimal cluster with probability at least $g(\epsilon)/2$, and crucially, this probability is independent of the number of unsettled optimal clusters. Once an unsettled optimal cluster becomes settled with $180\log n$ points sampled from its inner ring, it stops contributing to the probability of sampling a point from the inner rings of unsettled optimal clusters. Subsequently, points sampled in C with probability at least $g(\epsilon)/2$ would account for the number of points sampled from the inner rings of the remaining unsettled optimal clusters.

Since a sampled set C of size $\frac{4320k\log n}{g(\epsilon)}$ contains $180k\log n$ points from the inner rings of unsettled optimal clusters with probability at least $(1-1/n^k)$, the probability that all optimal clusters become settled using sampled set C is at least $(1-1/n^k)(1-1/n^2)$. □

Lemma F.10. *Let $C\subset X$ be a set of $\frac{4320}{g(\epsilon)}\cdot k\log n$ sampled points. Then, we obtain $\tilde{\Phi}(X,C)\le 40(1+\epsilon)\cdot OPT$ with probability at least $(1-1/n^k)(1-1/n^2)(1-1/n)$.*

Proof. Using Lemma F.8, we know that if we sample a center set C of size $\frac{4320}{g(\epsilon)}k\log n$ following the $\widetilde{D^2}$-sampling distribution, the probability that C contains at least $180k\log n$ points from the inner rings of the optimal clusters is at least $1-\frac{1}{n^k}$. Following Lemma F.9, for a sampled set C of size at most $\frac{4320}{g(\epsilon)}\cdot k\log n$, all clusters become settled with probability at least $(1-1/n^k)(1-1/n^2)$. Using Lemma F.3, the probability that event $\mathcal{E}$ holds for all iterations of Algorithm 3 is at least $(1-1/n)$. Hence, Algorithm 3 gives a $40(1+\epsilon)$-approximation for k-means with probability at least $(1-1/n^k)(1-1/n^2)(1-1/n)$.

Query complexity: We initiate the algorithm with $180\log n$ centers, and at the end of the algorithm, we have in total $180\log n+\frac{4320\cdot 29160}{\epsilon^3}\cdot k\log n=O\left(\frac{k\log n}{\epsilon^3}\right)$ centers. In every iteration, we sample a center, and for that center, we query the weak oracle $\widetilde{WO}$ for every $x\in X$ to get the distances. Hence, we make n weak-oracle queries in each iteration. Since there are $O\left(\frac{k\log n}{\epsilon^3}\right)$ iterations, in total, we make $O\left(\frac{nk\log n}{\epsilon^3}\right)$ queries to the weak-oracle. To compute $d^{est}_{km}(x,C_i)$ for all i in Algorithm 3, we need to perform strong-oracle queries. We make these strong-oracle queries to find the distances between any two centers in C_i. Hence, the number of strong-oracle queries required by Algorithm 3 is at most $O(\frac{k^2\log^2 n}{\epsilon^6})$.

With probability at least $(1-1/n^2)$, we sample at most $180\log n$ points from the inner ring of any unsettled optimal cluster A. We require the number of points in the inner ring, $(1-1/\alpha^2)|A|$, to be at least $180\log n$. This gives a lower bound on the size of each optimal cluster A of $\frac{480\log n}{\epsilon}$, for $\alpha=1+\epsilon/3$. □

F.2 Removing Lower Bound Assumption on Cluster Sizes

In Section F, we showed that conditioned on each optimal cluster having at least $\frac{480\log n}{\epsilon}$ points, there exists an algorithm for k-means clustering that gives a $40(1+\epsilon)$-approximation with high probability. In this section, we describe how do adapt the algorithm when there are optimal clusters with fewer than $\frac{480\log n}{\epsilon}$ points. These are the optimal clusters A for which the inner ring $IR(A,\alpha)$ contains fewer than $180\log n$ points in it. We know that following Lemma F.7 and Lemma F.8, points from the inner ring of an unsettled optimal cluster are sampled with probability at least $g(\epsilon)/2$ and once we sample $\frac{4320}{g(\epsilon)}\cdot k\log n$ points, we sample $180\log n$ points from the inner rings of all optimal clusters. Now, if there are less than $180\log n$ points in the inner ring of some optimal clusters, it implies that we sample all points of those inner rings. However, since the number of points sampled from the inner ring is less than $180\log n$, we cannot apply Lemma F.5 to claim that the unsettled optimal cluster A would become settled.

We modify our algorithm as follows. Since there might exist small clusters with less than $\frac{480\log n}{\epsilon}$ points, we increase the total number of points to be sampled to ensure that $\frac{480\log n}{\epsilon}$ points will be sampled from any

unsettled optimal cluster. The set C of samples will have size $O(\frac{k \log n}{\epsilon g(\epsilon)})$. For optimal clusters with more than $180 \log n$ points in their inner ring, as before, the Algorithm 3 will sample $180 \log n$ points from their inner rings and those clusters will become settled with high probability. For optimal clusters with fewer than $180 \log n$ points in their inner rings, either all points of those clusters will be sampled or the cluster will become settled. Substituting $g(\epsilon) = O(\epsilon^3)$, we get the sample complexity of this modified algorithm to be $O(\frac{k \log n}{\epsilon^4})$, and for the same approximation guarantee of $40(1+\epsilon)$-approximation, the number of strong-oracle and weak-oracle queries will be $O(\frac{k^2 \log^2 n}{\epsilon^8})$ and $O(\frac{nk \log n}{\epsilon^4})$, respectively.

Theorem F.2. *Let $\epsilon \in (0,1)$ and $\delta \leq 1/3$. There exists a randomized algorithm for k-means that makes $O(\frac{k^2 \log^2 n}{\epsilon^8})$ strong-oracle queries and $O(\frac{nk \log n}{\epsilon^4})$ weak-oracle queries to yield a $(O(\frac{\log n}{\epsilon^4}, 40(1+\epsilon)))$ bi-criteria approximation for k-means, and succeeds with at least some constant probability.*

Remark F.1. *We note that in Algorithm 3, the constant i.e., $4320 \cdot 29160$ appearing in the value of t, before $\frac{k \log n}{\epsilon^3}$, can be improved by using the biased Cauchy–Schwarz inequality (Fact A.1) in place of the generalized triangle inequality in the analysis above. This change affects only the hidden constant in the big-O notation and does not affect the statement of Theorem F.2 for k-means. In the following, we indicate exactly where the biased Cauchy–Schwarz inequality should replace the generalized triangle inequality to achieve this improvement.*

Using the generalized triangle inequality in equation 1, the cost of an optimal cluster A with respect to the current center set C_i satisfies $\tilde{\Phi}(A, C_i) \leq 2(1+9\alpha^2) \cdot OPT(A)$. On the other hand, applying the biased Cauchy-Schwarz inequality gives $\tilde{\Phi}(A, C_i) \leq (\beta+1)\left(\frac{1}{\beta} + 9\alpha^2\right) \cdot OPT(A)$. For $\beta = 1$, this recovers the same bound $\tilde{\Phi}(A, C_i) \leq 2(1+9\alpha^2) \cdot OPT(A)$, but choosing $\beta < 1$ yields a strictly improved bound. Next, using the Cauchy–Schwarz inequality in Equation 2 and substituting the resulting bound on $\tilde{\Phi}(A, C_i)$ into Equation 3 gives $v > \sqrt{\frac{20(1+\epsilon)-(\beta+1)\beta}{9(\beta+1)}}\, r$. When $\beta = 1$, this simplifies to $v > \sqrt{1 + \frac{10\epsilon}{9}}\, r$, which recovers the earlier bound for v in Equation 4. If we instead set $\beta = 0.15$, then the above expression becomes $v > \sqrt{1 + \frac{20\epsilon}{10.35}}\, r$. We now choose $\alpha = \left(1 + \frac{2\epsilon}{3}\right)$. Substituting the chosen values of β and α into the new bound for $\tilde{\Phi}(A, C_i)$ gives $\tilde{\Phi}(A, C_i) \leq 20(1+\epsilon) \cdot OPT(A)$. Recall that when sampling a point according to the $\widetilde{D^2}$ distribution, the probability that a sampled point x lies in the inner ring of an unsettled cluster A satisfies $\Pr(x \in IR(A, \alpha) \mid x \in A, A \in B_i) \geq \frac{(1-1/\alpha^2)(v-\alpha r)^2}{18(r^2+9v^2)} = \frac{(1-1/\alpha^2)(v/r-\alpha)^2}{18(1+9v^2/r^2)}$. Substituting the chosen values of α and v, we obtain $\Pr(x \in IR(A, \alpha) \mid x \in A, A \in B_i) \geq \left(1 - \frac{1}{\left(1+\frac{2\epsilon}{3}\right)^2}\right) \frac{\left(\sqrt{1+\frac{20\epsilon}{10.35}} - \left(1+\frac{2\epsilon}{3}\right)\right)^2}{180(1+18\epsilon/10.35)} > \frac{\epsilon^3}{13767}$. Since this probability is larger than what we had earlier, fewer centers need to be sampled, and consequently the required value of t decreases to $t = 4320 \cdot 13767$, where it was $t = 4320 \cdot 29160$ earlier.

F.3 Constant Approximation for k-means in Weak-strong Oracle Model

Theorem F.3. *There exists a randomized algorithm for k-means in the weak-strong oracle model that computes a constant factor approximation with constant probability using $O(\frac{k^2 \log^2 n}{\epsilon^8})$ strong-oracle queries and $O(\frac{nk \log n}{\epsilon^4})$ weak-oracle queries.*

Proof. We use the algorithm of Theorem F.2 to obtain a bi-criteria approximation guarantee for k-means. Let C_{t+1} denote the resulting set of centers. Each center $c_i \in C_{t+1}$ is initialized with weight $w(c_i) = 0$. Next, we assign every point x to the center in C_{t+1} for which $d_{km}^{est}(x, C_{t+1})$ is minimized. This create a weighted k-means instance with $O(\frac{k \log n}{\epsilon^4})$ points where each center c_i in C_{t+1} is assigned $w(c_i)$ points. Since we have used strong-oracle queries for all pairs of points in this weighted instance, we know the exact distances between all pairs of points. We run a constant factor approximation algorithm, such as the algorithm by Mettu and Plaxton (2004) on this weighted k-means instance to obtain a constant factor approximation for k-means. The constant factor approximation guarantee for k-means follows using Proposition 20 of Bateni et al. (2024). □

G LOWER BOUND IN THE WEAK-ORACLE MODEL

In this section we show that any randomized algorithm giving a constant factor approximation for the k-means problem in the weak-oracle model requires to make expected $\Omega(\frac{nk}{(1-2\delta)^2})$ queries. For simplicity of calculations, we show this result for the k-median problem. Similar results follow for the k-means problem as well.

Theorem G.1. *Let $\delta \in (0, 1/2)$. Any randomized algorithm in the weak-oracle model giving any constant factor approximation for the k-median problem with probability at least $3/4$ must make $\Omega(\frac{nk}{(1-2\delta)^2})$ of queries in expectation to the weak-oracle.*

Proof. To prove the lower bound, we construct a hard instance for which any randomized algorithm giving any constant factor approximation for k-median with high probability requires to make a lot of weak-oracle queries. Consider an instance X with n points. These n points are partitioned into k groups, each containing $\frac{n}{k}$ points. We denote these partitions as $C_1, C_2, \ldots, C_k$. We define distances between points in X as follows: for any two points x, y belonging to the same partition C_i, we set $d(x, y) = 1$, and for points $x \in C_i$ and $y \in C_j$, where $i \neq j$, we set $d(x, y) = l$, where l is a large number. These distances satisfy the metric property. For this instance, the optimal k-median solution has cost $n - k$.

Recall that an algorithm for k-median in the weak-oracle model gets access to distance values only through the weak-oracle. Suppose that all points in X have been clustered correctly except for one point, v. We need to assign v to its correct cluster, say C_v. If v gets assigned to an incorrect cluster, then the cost for v would be l. Hence, the k-median clustering cost would be at least $(n - k - 1) + l$. For $l = c \cdot (n - k)$ for some constant $c > 0$, the k-median cost of a clustering that misclassifies even a single point is at least some constant times the value of the optimal solution. Therefore, the algorithm needs to correctly classify all input points to achieve a constant factor approximation for k-median.

Let $\bar{d}(x, y)$ denote the value returned by the weak-oracle for any two input points x, y. Then, the following holds.

$$\bar{d}(x,y) = \begin{cases} 1 & \text{with probability } 1-\delta \text{ if } \ x, y \text{ belong to the same cluster} \\ l & \text{with probability } \delta \text{ if } \ x, y \text{ belong to the same cluster} \\ l & \text{with probability } 1-\delta \text{ if } \ x, y \text{ belong to different clusters} \\ 1 & \text{with probability } \delta \text{ if } \ x, y \text{ belong to different clusters} \end{cases}$$

For our query lower bound in the weak-oracle model, we use a lower bound result for the Query-Cluster problem studied by Mazumdar and Saha (2017). The authors consider a setup where there is an unknown ground truth k-clustering of a set X of n points, and the algorithmic task is to recover the ground truth clustering using queries to an oracle that answers whether any two points belong to the same cluster or not. Moreover, the query answers are wrong independently with probability $\delta < 1/2$. The authors prove a $\Omega(\frac{nk}{(1-2\delta)^2})$ lower bound on the number of oracle queries for this Query-Cluster problem that recovers the ground truth k-clustering.

Our weak-oracle query lower bound of $\Omega(\frac{nk}{(1-2\delta)^2})$ follows by observing that one can reduce the Query-Cluster problem to the cluster recovery problem on an instance generated as above by the weak-oracle. Hence, the weak-oracle query lower bound $\Omega(\frac{nk}{(1-2\delta)^2})$ holds for any algorithm giving any constant factor approximation of k-median in weak-oracle model. □

H ALGORITHM FOR K-CENTER IN WEAK-STRONG ORACLE MODEL

We start by exploring whether known algorithms for the k-center problem can be implemented in the weak-strong oracle model. One key challenge is the *assignment* of points to centers. This means that even if we can locate k good centers, we also need to output a mapping of points to the nearest center. This implies that we need to be able to calculate the distances of every point to a set of centers. Since we can use only a sublinear number of strong-oracle queries, we do not have accurate distances for doing this mapping. The lack of accurate distances also comes in the way of finding good centers since mapping and finding good centers go hand-in-hand in k-center algorithms. So, the important question is: *Can we get a reasonably accurate estimate of distances using weak-oracle queries?* We faced similar issues while designing an algorithm for the k-means problem as well. We will show that the distance function as defined in Definition F.1 will help in designing an algorithm for the k-center problem as well.

We will design a randomized algorithm within the weak-strong model that gives a $(6 + \varepsilon)$-approximation for the k-center problem. The main idea of the algorithm can be better understood in a simpler setting where the accurate distances are known. Let us see the outline of this algorithm. We shall assume that the optimal k-center

radius r_{opt} is known.[1] The algorithm is a simple "*greedy ball-carving*" procedure that is commonly used in the context of the k-center problem and can be stated as the following pseudo code (also stated as Algorithm 3 in Bateni et al. (2024)):

Algorithm 4: `Greedy Ball Carving`

Input : Set of points S, radius Rad
Output: A set of $C = \{c_1, \ldots, c_m\}$ centers and assignment of points in S to centers in C
▷ Initialize: $C = \{\}$
while S *is not empty* **do**
  ▷ Pick an arbitrary point $c \in S$
  ▷ Treat c as a center and assign any point $s \in S$ to c if $d(c, s) \leq 2Rad$.
  ▷ Add c to C and remove all points assigned to c from S.
end

We execute the above algorithm with inputs $S = X$ and $Rad = r_{opt}$. We can argue that the greedy ball carving algorithm terminates after picking at most k centers and assigning all points in X. This means that the algorithm gives a 2-factor approximation guarantee.

Lemma H.1. *Let (X, k) be any instance of the k-center problem and let r_{opt} denote the optimal k-center radius for X. The greedy ball carving procedure, when executed with inputs $S = X; Rad = r_{opt}$, gives a 2-approximate solution for the instance (X, k).*

Proof. Let $\{c_1^*, \ldots, c_k^*\}$ be the optimal k centers and let $X_1^*, \ldots, X_k^*$ be the corresponding k clusters of dataset X. So, for every i, $X_i^* \subset \mathcal{B}(c_i^*, r_{opt})$. If the k centers chosen by the greedy ball carving algorithm belong to k distinct optimal clusters $X_1^*, \ldots, X_k^*$ and we assign points by carving out balls of radius $2r_{opt}$ around every chosen center, then all points are guaranteed to get assigned to a center within a distance $2r_{opt}$, giving a 2-factor approximation. So, all we need to show is that the algorithm chooses centers from all of the distinct optimal clusters. We can show this by induction on the iteration number. The statement trivially holds for the first iteration. Suppose the algorithm chooses centers from distinct clusters $j_1, \ldots, j_i$ in the first i iterations. Note that in the next iterations, the algorithm picks centers from the set of points that does not contain any of the points in $X_{j_1}^*, \ldots, X_{j_i}^*$ as they have been carved out when removing points within balls of radius $2r_{opt}$ around the chosen centers. So, the $(i+1)^{th}$ center will be chosen from a new optimal cluster. So, the ball carving algorithm terminates after picking k centers from k distinct optimal clusters. This completes the proof of the lemma. □

The reason we discussed the greedy ball carving algorithm and its analysis is that our $6(1+\varepsilon)$-factor approximation follows the same template. The only complication is that during the ball carving step, where we remove all points within the radius $2r_{opt}$, we need accurate distances, which we do not have in the weak-strong oracle model. So, we need to do more than choose and center, and carve out a ball of radius $2r_{opt}$ in every step. Let us continue assuming that the optimal radius r_{opt} is known. We will give a 6-factor approximation under this assumption, which changes to $6(1+\varepsilon)$ when the assumption is dropped. To enable distance estimates, along with a center c_i, we must also pick a set of nearby points to c_i, which will help in estimating the distance of any point y to c_i following Lemma F.1. A reasonable way to do this would be to uniformly sample a set T_i of points (instead of one), use the strong-oracle queries on the subset and pick a center $c_i \in T_i$ with the largest number of points in T_i within a radius $2r_{opt}$. Let S_{c_i} denote the subset of points in T_i within a distance of $2r_{opt}$ to c_i. We can now obtain distance estimates using the tuple (c_i, S_{c_i}) using the weak-oracle queries as in Lemma F.1 and carve out those points and assign them to c_i that satisfy $d_{km}^{est}(c_i, z) \leq r$ for an appropriate value of r. Let c_i belong to the optimal cluster X_i^*. What should the appropriate value of r be such that the points in $S \cap X_i^*$ are guaranteed to get carved out and assigned to c_i? Since the distance estimates are inaccurate, we must use $r = 4r_{opt}$. However, this may cause a point that is $6r_{opt}$ away from c_i to get carved out and assigned to c_i. This is the reason we obtain an approximation guarantee of 6. This high-level analysis lacks two relevant details: (i) how do we eliminate the assumption that r_{opt} is known, and (ii) probability analysis for all points being assigned a center within distance $6r_{opt}$. We remove the assumption about r_{opt} being known by iterating over discrete choices for the optimal radius $Rad = (1+\varepsilon)^0, (1+\varepsilon), (1+\varepsilon)^2, \ldots$. We will argue that the ball carving succeeds for the choice $Rad = r_{opt}(1+\varepsilon)$ with high probability. So, the approximation guarantee we get is $6(1+\varepsilon)$. Instead of iterating

[1] This is not an unreasonable assumption to make since we can guess the optimal radius with $(1 \pm \varepsilon)$ by iterating over discrete choices of the radius.

over the possible choices of Rad linearly, we can do that using binary search, which results in the running time and query complexity getting multiplied by a factor of $O\left(\log \frac{\log \Delta}{\varepsilon}\right)$ due to this repeated ball carving (once for every choice of Rad), where Δ is the aspect ratio. Assuming Δ to be polynomially bounded (which is a popular assumption), the multiplicative factor becomes $O\left(\log \frac{\log n}{\varepsilon}\right)$. In the remaining discussion, we state our algorithm more precisely and do the probabilistic analysis.

Algorithm 5: `Weak-Greedy Ball Carving`

Input : Set of points S, radius Rad
Output: A set of $C = \{c_1, \ldots, c_m\}$ centers and assignment of points in S to centers in C
▷ Initialize: $C = \{\}$
while *S is not empty* **do**
  (1) If ($|C| = k$) **abort**
  (2) If ($|S| \leq 180k \log n$), use strong-oracle queries to find the remaining centers covering all points in S within distance $2Rad$. If this is not possible, **abort**. Otherwise, output the k centers picked.
  (3) Pick a subset $T \subset S$ of size $180k \log(n)$ uniformly at random
  (4) Query the strong-oracle to find distances between elements in T and use these to find a center $c \in T$ such that $|S_c|$ is maximized, where $S_c \equiv |\mathcal{B}(c, 2Rad) \cap T|$.
  (5) If ($|S_c| < 180 \log n$) **abort**
  (6) Retain $180 \log n$ points in S_c (*i.e., remove the extra point*)
  (7) Treat c as a center and assign any point $s \in S$ to c if $d^{est}_{km}(s, c) \leq 4Rad$.
  (8) Add c to C and remove all points assigned to c from S.
end
return *C and the assignment*

We prove the following lemma related to Algorithm 5.

Lemma H.2. *Let (X, k) be any instance of the k-center problem and let r_{opt} denote the optimal k-center radius for X. Then, Algorithm 5 satisfies the following two properties:*

1. *For any value of Rad, given that the algorithm does not abort, the probability that all points are within $6Rad$ of the centers chosen is at least $(1 - 1/n^4)$.*

2. *For any $Rad \geq (1+\varepsilon)r_{opt}$, the probability that the algorithm does not abort is at least $(1 - 1/n^4)$.*

Proof. For the second property, we will use the same induction-based argument as in the proof of Lemma H.1. Given that in the first $i-1$ iterations, the centers $c_1, \ldots, c_{i-1}$ belong to $i-1$ distinct optimal clusters $X^*_{j_1}, \ldots, X^*_{j_{i-1}}$ and S does not contain any points from these clusters, we will show that the probability that the next chosen center c_i belonging to a new cluster $X^*_{j_i}$ carves out any point $z \in S \cap X^*_{j_i}$ is at least $(1 - 1/n^5)$. Note that by our choice of center c_i and the fact that $Rad \geq (1+\varepsilon)r_{opt}$, we have $|S_{c_i}| \geq 180 \log n$. Consider any point $z \in X^*_{j_i} \cap S$. We have $d(z, c_i) \leq 2Rad$ and from Lemma F.1, $\Pr[|d^{est}_{km}(z, c_i) - d(z, c_i)| \leq 2Rad] \geq 1 - \frac{1}{n^5}$. From this we get $\Pr[d^{est}_{km}(z, c_i) \leq 4Rad] \geq 1 - \frac{1}{n^5}$. So, with probability at least $(1 - \frac{1}{n^5})$, z gets assigned to the center c_i. So, the probability that there is a point in $X^*_{j_i} \cap S$ that does not get assigned to c_i is at most $\frac{1}{n^5}$. So, by union bound, the probability of a point in X that does not get assigned to centers $c_1, \ldots, c_k$ is at most $\frac{1}{n^4}$.

For the first property, let us bound the probability that a point z gets assigned a center that is at a distance $> 6Rad$. Suppose the z is assigned a center in iteration i. So, it gets assigned center c_i. This means that $d^{est}_{km}(z, c_i) \leq 4Rad$. The probability that $d(z, c_i) > 6Rad$ is at most $\frac{1}{n^4}$ from the previous discussion. So, the probability that there is a point that gets assigned a center which is $> 6Rad$ distance away is at most $\frac{1}{n^4}$ from the union bound. So, property (1) follows. □

Let us see why calling Algorithm 5 with input (X, Rad) while doing a binary search over Rad gives a $6(1+\varepsilon)$-approximate solution with high probability. From Lemma H.2, if $Rad < (1+\varepsilon)r_{opt}$, and the algorithm does not abort, then with probability at least $(1 - 1/n^4)$ all points are assigned a center that is within distance $6Rad < 6(1+\varepsilon)r_{opt}$. On the other hand, if $Rad \geq (1+\varepsilon)r_{opt}$, then the algorithm does not abort with probability at least $(1-1/n^4)$. So, the binary search will terminate with a $6(1+\varepsilon)$-approximate solution with high probability.

Query complexity: The number of strong-oracle queries in every iteration of the `weak greedy ball carving` algorithm is $O(k^2 \log^2 n)$. So, the total number of strong queries across k iterations is $O(k^3 \log^2 n)$. The number of weak Oracle queries is $O(nk \log n)$. Taking into account the binary search over Rad, the number of strong-oracle and weak-oracle queries are $O\left(k^3 \log^2 n \log \frac{\log n}{\varepsilon}\right)$ and $O\left(nk \log n \log \frac{\log n}{\varepsilon}\right)$, respectively.

Theorem H.1. *There exists a $6(1+\epsilon)$-approximation algorithm for the k-center problem that makes $O\left(k^3 \log^2 n \log \frac{\log n}{\varepsilon}\right)$ strong-oracle and $O\left(nk \log n \log \frac{\log n}{\varepsilon}\right)$ weak-oracle queries and succeeds with probability at least $(1-1/n^4)^2$.*

Remark H.1. *We highlight that for $k \ll \log n$, one can obtain better strong-oracle query bounds for k-center clustering. We sketch the main idea below. We sample a set T of $O(k \log n)$ points uniformly at random. Next, we subsample T to obtain Y, where each $x \in T$ is included in Y independently with probability $1/k$. Using linearity of expectation, we have $\mathbb{E}[|Y|] = O(\log n)$. Consider any large optimal cluster C^* such that sample T contains at least $O(\log n)$ points of that cluster. Since $k \ll \log n$, with high probability, at least one point from the optimal cluster C^* will be sampled in Y. We think of the set Y as the set of* good *centers. Since $\mathbb{E}[|Y|] = O(\log n)$ and for each $y \in Y$, we make strong-oracle queries to all $x \in T$, overall this makes $O(k \log^2 n)$ strong-oracle queries in expectation. The algorithm repeats the above step at most k times. We repeat the above steps at most $O(\log \frac{\log n}{\epsilon})$ times for the guess of the optimal radius Rad. Hence, overall the algorithm makes at most $O(k^2 \log^2 n \log \frac{\log n}{\epsilon})$ strong-oracle queries in expectation. Using standard concentration bounds, one can show that the algorithm makes $O(k^2 \log^2 n \log \frac{\log n}{\epsilon})$ strong-oracle queries with high probability as well.*

I EXPERIMENTAL RESULTS

We run our algorithms for k-means and k-center problems on synthetic as well as real-world datasets to demonstrate that our algorithms can be implemented efficiently in practice, and in particular, our algorithms outperform the algorithms of Bateni et al. (2024). In our experiments, we vary the failure probability δ of the weak-oracle and run our algorithms on datasets of different sizes, and report how the cost of clustering varies with the number of oracle queries.

Remark I.1. *In our discussions of strong-oracle queries so far, we interpret a strong-oracle query as a query for the distance between two points. We note that Bateni et al. (2024), in their experiment results, report the number of* point *queries, where an oracle on a point query provides the embedding of the point. Consider the following example for more clarity about the relation between the number of point queries and the number of strong-oracle queries. Let us have a set S of points, and let us make a point query for each of the $|S|$ points. This will give us the embedding of all points in the set S, and using these embeddings, we can compute the pairwise distances between all pairs of points in S. On the other hand, to find the pairwise distances between all pairs of points in S, one must make $\binom{|S|}{2}$ strong-oracle queries.*

We note that Bateni et al. (2024) reports their experimental results using the number of point queries. We convert the reported point query counts to strong-oracle query counts using the relationship mentioned above, and compare the number of strong-oracle queries used in Bateni et al. (2024) with those obtained in our experiments.

Experimental setup The initial experiments were carried out on a Mac Mini with an M2 chip and 8GB RAM. Subsequently, the computationally intensive experiments were performed on a server having 1.5 TB RAM with 64 CPU cores. It took about 10 days to obtain the experimental results.

Datasets We run our algorithms on synthetic as well as real-world datasets, and compare the performance of our algorithms with those of Bateni et al. (2024). The datasets used in our experiments are described as follows. We generate synthetic data using the Stochastic Block Model (SBM) (Mossel et al., 2015; Abbe, 2018). We use three synthetic datasets in our experiments with the number of points n being $10k, 20k$ and 50k, respectively. Each of these datasets has $k = 7$ clusters, and the points in the datasets belong to a seven-dimensional space. The points in the ith cluster of the datasets are drawn from a Gaussian distribution $N(\mu^{(i)}, I)$, where $\mu^{(i)}[i] = 10^5$ and $\mu^{(i)}[j] = 0$ for $j \neq i$. This ensures that the points within the same cluster are close to each other, while the optimal clusters remain well separated.

For the real-world data, we use the MNIST dataset (Deng, 2012; Van der Maaten and Hinton, 2008). The MNIST dataset has ten clusters, $k = 10$. The dataset has $n = 60k$ images, each with a dimension of 28×28. Similar to

Bateni et al. (2024), we apply SVD and t-SNE embeddings to embed the dataset into $d = 50$ dimensions and $d = 2$ dimensions, respectively.

Construction of the weak-oracle In order to construct the weak-oracle, we first create a perturbed distance matrix for both datasets as follows. This matrix is initialized with the actual distances between the points in the dataset. For the SBM-based synthetic dataset, for any two points i and j, if they belong to the same cluster, independently with probability δ, the (i, j)th entry of the perturbed matrix is set to 10^5. And if i and j belong to different clusters, independently with probability δ, the (i, j)th entry of the matrix is set to one. This is how we create the perturbed matrix using which the weak-oracle responds to a query for the SBM-based synthetic dataset.

For the MNIST dataset, if two points belong to the same cluster (i.e., the same digit class), we replace the distance between the points in the matrix with a randomly chosen inter-cluster distance between two points in different clusters. If the points belong to different clusters (i.e., different digit classes), we assign them a randomly chosen intra-cluster distance. Bateni et al. (2024) notes that the SBM and MNIST datasets embedded using t-SNE with $d = 2$ and SVD with $d = 50$, have clear ground-truth clusterings. Since we know the ground-truth clustering for these embeddings, we can construct the weak-oracle in the manner described above.

Baselines in our experiments We run the k-means++ algorithm on these datasets using the strong-oracle, and use its output as the *strong baseline* in our experiments for k-means. Similarly, we run the k-means++ algorithm on these datasets using the weak oracle, and use its output as the *weak baseline*. In our experimental results, we compute the approximation factor of our algorithm by calculating the ratio of the cost of our algorithm with this strong baseline.

We compare the performance of our algorithms for k-center with the baselines computed as follows. The output of the farthest point algorithm of Gonzalez (1985) gives us the baseline for both the SBM-based synthetic dataset as well as the MNIST dataset. The strong and weak baselines for the experiments on k-center are obtained by running the algorithm of Gonzalez (1985) with the strong-oracle and weak-oracle, respectively. In the experiments, we report the approximation factors obtained by our algorithms as the ratio of the cost of our algorithm with the strong baseline.

Experimental results on k-means clustering In our experiments, we run our k-means algorithm on SBM-based synthetic datasets of size $n = 10k, 20k$ and $50k$, and set the failure probability δ of the weak-oracle to be $0.1, 0.2$ and 0.3. For each of the above choices, we run our algorithm with the number of strong-oracle queries, decided as follows. Using our theoretical results, $O\left(\frac{k^2 \log^2 n}{(1/2-\delta)^2}\right)$ strong-oracle queries suffice to get a constant approximation. In the experiments, we vary the constant factor in the expression $O\left(\frac{k^2 \log^2 n}{(1/2-\delta)^2}\right)$ from 0.0001 to 10 and observe the resulting clustering cost.

Table 4: Performance of our k-means algorithm on MNIST dataset

MNIST	% of strong-oracle queries (Ours)			Approximation factor			% of strong-oracle queries (Bateni et al., 2024)			% dropped
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	Value
SVD	0.00013	0.00019	0.00024	1.019	1.03	1.027	0.00062	0.00096	0.00063	61
t-SNE	0.00054	0.00079	0.00089	1.1595	1.1433	1.0960	0.20969	0.20877	0.43814	99

Table 5: Performance of our k-means algorithm on SBM-based synthetic data

n	% of strong-oracle queries (Ours)			Approximation factor			% of strong-oracle queries (Bateni et al., 2024)			% dropped
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	Value
10k	0.00585	0.00616	0.00616	2.156	0.9664	0.998	0.3075	0.12286	1.73861	94
20k	0.00200	0.00218	0.00242	1.065	1.089	1.0695	0.03890	0.03159	0.73405	93
50k	0.00039	0.00043	0.00045	1.043	1.058	1.015	0.01075	0.00670	0.05480	93

Tables 4 and 5 show our experimental results for k-means clustering on the MNIST dataset with SVD and t-SNE embeddings, and on SBM-based datasets of sizes $n = 10k$, $20k$, and $50k$, and present a comparison of the percentage of strong-oracle queries used by our algorithms and that of Bateni et al. (2024). We conducted experiments for different combinations of failure probability δ and the number of strong-oracle queries. In Table 5, we mention the results for the setting for which the value of (number of strong-oracle queries $\times$ log(cost of clustering)) is minimized.

Comparison of performance of our k-means algorithm with Bateni et al. (2024): We compare the performance of our k-means algorithm with the k-means algorithm of Bateni et al. (2024) as follows. Figures 1, 2 and 3 capture how the clustering cost varies with the number of strong-oracle queries when these algorithms are run on SBM-based synthetic datasets of size $10k$, $20k$ and $50k$, respectively. The blue, green, and red lines in the plots represent results for $\delta = 0.1$, $\delta = 0.2$, and $\delta = 0.3$, respectively. The solid lines show how the clustering cost behaves as the number of strong-oracle queries is changed in our algorithm. The dotted lines represent the results from Bateni et al. (2024). In Figures 1, 2, and 3, the solid blue and green lines are not visible because they are hidden behind the red lines. This happens because, for all values of δ, our percentage of strong oracle queries stays around 0.005, which is much lower compared to the number of strong oracle queries used in Bateni et al. (2024). As a result, all the blue and green lines appear overlapped by the red lines. Note that since we don't have their exact data, we reported these values by visually interpreting their plots.

We observe in the experiments that as the number of strong-oracle queries increases, the clustering cost given by our algorithm as well as the algorithm in Bateni et al. (2024) decreases. Take any value of δ, say $\delta = 0.1$. The cost versus the number of strong-oracle queries behavior of our algorithm is represented by the solid blue line, while the performance of the algorithm of Bateni et al. (2024) is shown using the dotted blue line in Figures 1, 2 and 3. We observe that in both Figures 1, 2 and 3 our algorithm achieves a lower cost with significantly fewer strong-oracle queries. Similar observations hold for other values of δ. Our algorithm makes at least 93% fewer strong-oracle queries compared to Bateni et al. (2024) for every choice of $\delta \in \{0.1, 0.2, 0.3\}$ and $n \in \{10k, 20k, 50k\}$.

We note that the costs of k-means solutions of Bateni et al. (2024), given enough strong-oracle queries, as shown in Figures 1 and 2, tend to be smaller than the cost achieved by our k-means algorithm. It seems that the strong baseline used by Bateni et al. (2024) achieves a lower cost compared to the strong baseline used by our algorithm. Note that we use k-means++ with strong-oracle queries to compute the strong baseline for k-means.

Figure 1: k-means on SBM dataset with $n = 10k$

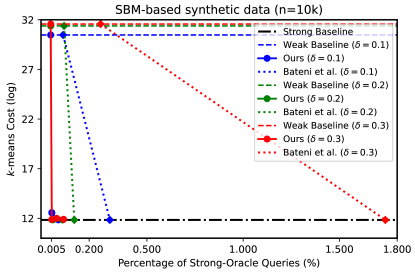


Figure 2: k-means on SBM dataset with $n = 20k$

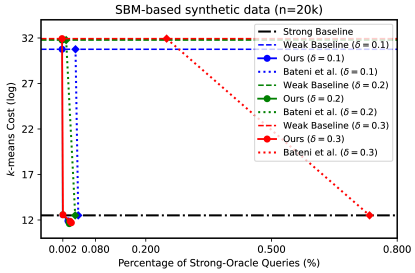
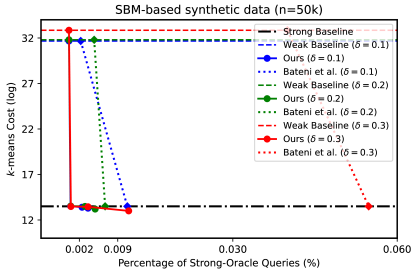


Figure 3: k-means on SBM dataset with $n = 50k$



Figures 4 and 5 show how our k-means clustering cost changes with respect to the number of strong-oracle

queries on the MNIST dataset using SVD and t-SNE embeddings. Unlike SBM-based datasets, the distances between clusters in the MNIST are relatively smaller. Because of this, there is little difference between the weak and strong baselines in these figures. Recall that to construct the weak oracle, if two points belong to the same cluster, we replace their distance with the distance between two points from different clusters. Since the inter-cluster distances in MNIST (with both SVD and t-SNE embeddings) are small, the weak and strong baselines end up being quite similar.

We believe that the k-means++ cost on the MNIST dataset using t-SNE embeddings is higher than the cost using SVD embeddings. However, based on the visualizations of k-means results on MNIST provided in Bateni et al. (2024), the baseline cost for t-SNE appears to be lower than that for SVD. Therefore, we could not directly use the results from Bateni et al. (2024) for comparison with our own.

Figure 4: k-means on MNIST with SVD embedding $n = 60k$

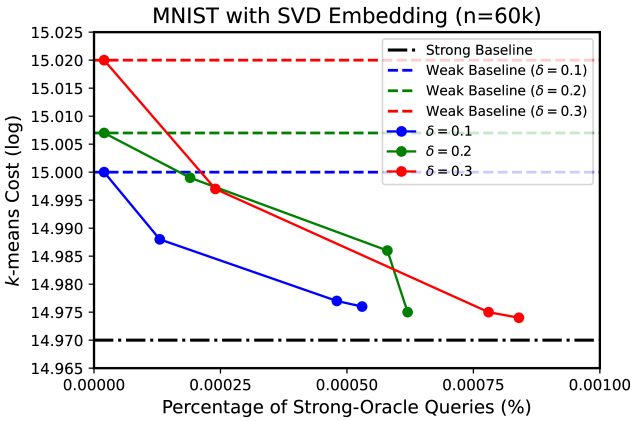
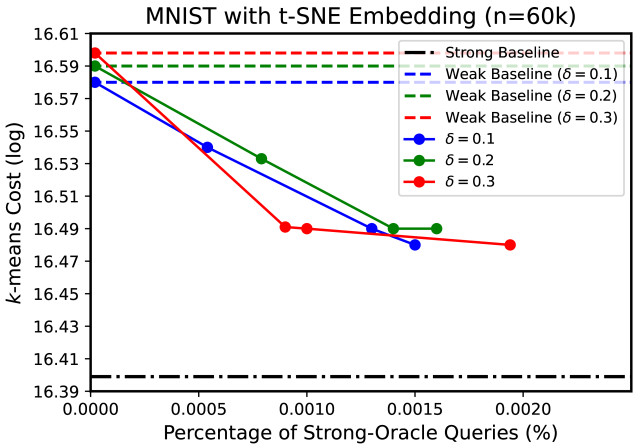


Figure 5: k-means on MNIST with t-SNE embedding $n = 60k$



Experimental results on k-center clustering We run our k-center algorithm on SBM-based synthetic datasets of size $n = 10k, 20k$ and $50k$ as well as the MNIST dataset with SVD and t-SNE embeddings, and vary the failure probability δ of the weak-oracle as $0.1, 0.2$ and 0.3. For each of the above choices, we run our k-center algorithm on the dataset with varied number of strong-oracle queries. The number of strong-oracle queries used in the experiments is decided as follows. Using our theoretical results, $\tilde{O}(\frac{k^3 \log^2 n}{(1/2-\delta)^2})$ strong-oracle queries suffice to get a constant approximation. In the experiments, we set the number of strong-oracle queries to be $O(\frac{k^3 \log^2 n}{(1/2-\delta)^2})$ and vary the constant factor in the above expression from 0.0001 to 10 and observe the cost of clustering mentioned below.

Table 6: Performance of our k-center algorithm on SBM-based synthetic data

n	% of strong-oracle queries (Ours)			Approximation factor			% of strong-oracle queries (Bateni et al., 2024)			% dropped
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	Value
10k	0.03980	0.06225	0.148625	0.84	0.9	0.86	0.42319	0.54987	0.54987	72
20k	0.01204	0.019531	0.04399	0.78	0.84	0.79	0.01581	0.03832	0.29785	23
50k	0.00192	0.00358	0.00808	0.81	0.8	0.77	0.00635	0.02199	0.04773	83

Table 7: Performance of our k-center algorithm on MNIST dataset

MNIST	% of strong-oracle queries			Approximation factor		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$
t-SNE	0.00062	0.00083	0.00089	2.46	2.45	2.31
SVD	0.00062	0.00084	0.00094	1.276	1.3	1.37

Tables 6 and 7 show our experimental results on k-center clustering on SBM-based synthetic datasets of size $n = 10k, 20k$ and $50k$ and on MNIST dataset with SVD and t-SNE embeddings and present comparison of the percentage of strong-oracle queries used by our algorithms and that of Bateni et al. (2024). We conducted experiments for different combinations of δ and the number of strong-oracle queries. In Table 6 and Table 7, we report the results for which the value of (number of strong-oracle queries $\times$ log(cost of clustering)) is minimized.

Comparison of performance of our k-center algorithm with Bateni et al. (2024): We compare the performance of our k center algorithm with the k-center algorithm of Bateni et al. (2024). Figures 6, 7, and 8 illustrate how the clustering cost changes as the number of strong-oracle queries increases. These results are based on SBM-based synthetic datasets of size $10k$, $20k$, and $50k$, respectively. In the plots, the blue, green, and red lines represent the results for $\delta = 0.1$, $\delta = 0.2$, and $\delta = 0.3$. Solid lines show the performance of our algorithm, while dotted lines indicate the results from Bateni et al. (2024). Since we do not have access to their exact data, we estimated their values by visually interpreting their plots. We observe significant reduction in the number of strong-oracle queries by our algorithm compared to Bateni et al. (2024). Our results show that our algorithms use at least $72\%, 23\%$ and 83% fewer strong-oracle compared to Bateni et al. (2024) for $n = 10k, 20k$ and $50k$, respectively.

Figure 6: k-center on SBM dataset with $n = 10k$

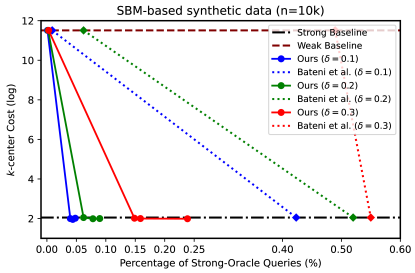


Figure 7: k-center on SBM dataset with $n = 20k$

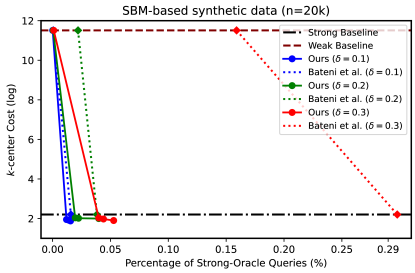
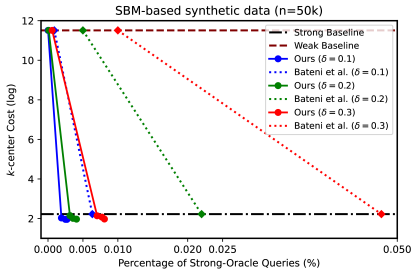


Figure 8: k-center on SBM dataset with $n = 50k$



Performance of k-center on MNIST dataset: Next, we describe the performance of our k-center algorithm on MNIST dataset with SVD and t-SNE embeddings. Figures 9 and 10 show how the clustering cost varies with the number of strong-oracle queries when these algorithms are run on MNIST datasets with SVD and t-SNE embeddings. We run these experiments for $\delta = 0.1, 0.2$ and 0.3, and the results are given as blue, green and red lines in the Figures 9 and 10. The dash-dotted line corresponds to the baseline computed using the algorithm of Gonzalez (1985). Bateni et al. (2024) does not provide any experimental results for the k-center problem on the MNIST dataset. In Figure 10, we observe that the k-center cost with $\delta = 0.3$ is lower than the costs for $\delta = 0.1$ and $\delta = 0.2$. We believe this may be due to the nature of the t-SNE embeddings, where the distances between clusters are relatively small. As a result, the distance between two points in the same cluster can be larger than the distance between two points in different clusters. Because of this, the weak oracle might return the distance between two points in the same cluster with a smaller distance.

Figure 9: k-center on MNIST with SVD embedding, $n = 60k$

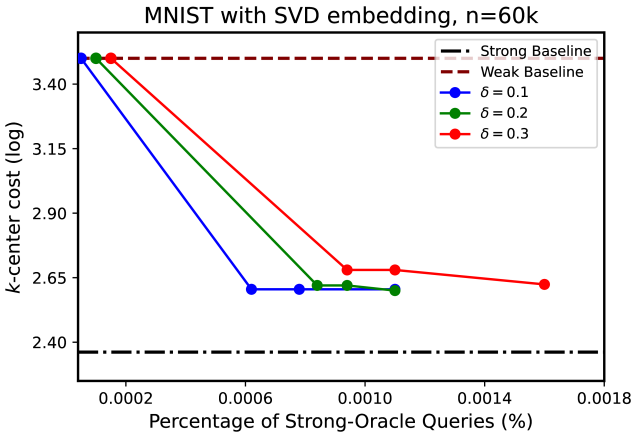


Figure 10: k-center on MNIST with t-SNE embedding, $n = 60k$

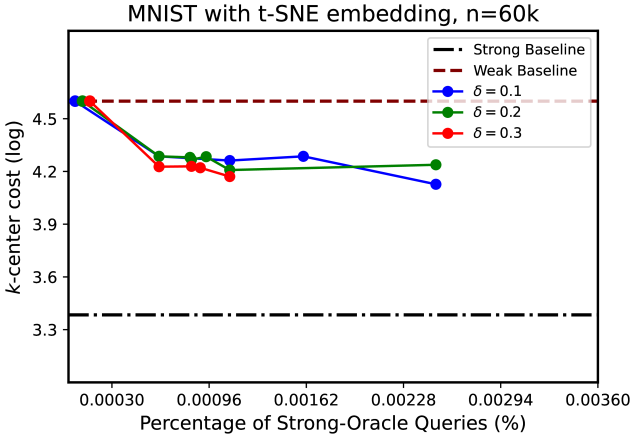


Table 8: Variance of experimental results for k-center and k-means clustering problems for different datasets and failure probability values. The values mentioned in the table are to be multiplied by 10^{-3} to obtain the actual variance results.

Datasets	k-center			k-means		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$
SBM, n=10k	5.1	3.2	5.1	51.4	5.9	0.77
SBM, n=20k	0	3.9	1.0	7.1	6.3	11
SBM, n=50k	7.8	3.9	7.8	12.7	11	7.5
MNIST, n=60k (t-SNE)	2.3	0	4.0	4.4	6.5	20.0
MNIST, n=60k (SVD)	3.2	35.0	8.4	0.3	1.3	1.0

Table 8 shows the variance results for the experiments we conducted. These results are computed as follows. For both the k-means and k-center problems, we first fix the dataset and the value of δ. Then, we vary the constant in the expressions used for the number of strong-oracle queries: $O\left(\frac{k^2 \log^2 n}{(1/2-\delta)^2}\right)$ for k-means, and $O\left(\frac{k^3 \log^2 n}{(1/2-\delta)^2}\right)$ for k-center. We try different constant values between 0.0001 and 10, and choose the value which minimizes the expression (number of strong-oracle queries) $\times$ log(cost of clustering). For each of these choices, we run our algorithm five times. The average of the clustering costs from these runs is reported as the clustering cost, and the variance across the five runs is shown in Table 8.