Proceedings of Machine Learning Research vol 247:1–53, 2024 37th Annual Conference on Learning Theory

Metric Clustering and MST with Strong and Weak Distance Oracles

MohammadHossein Bateni BATENI@GOOGLE.COM
Google Research

Prathamesh Dharangutte PRATHAMESH.D@RUTGERS.EDU
Rutgers University

Rajesh Jayaram RKJAYARAM@GOOGLE.COM
Google Research

Chen Wang CW200@RICE.EDU
Rice University and Texas A&M University

Editors: Shipra Agrawal and Aaron Roth

Abstract

We study optimization problems in a metric space $(\mathcal{X}, d)$ where we can compute distances in two ways: via a "strong" oracle that returns exact distances $d(x, y)$, and a "weak" oracle that returns distances $\tilde{d}(x, y)$ which may be arbitrarily corrupted with some probability. This model captures the increasingly common trade-off between employing both an expensive similarity model (e.g. a large-scale embedding model), and a less accurate but cheaper model. Hence, the goal is to make as few queries to the strong oracle as possible. We consider both "point queries", where the strong oracle is queried on a set of points $S \subset \mathcal{X}$ and returns $d(x, y)$ for all $x, y \in S$, and "edge queries" where it is queried for individual distances $d(x, y)$.

Our main contributions are optimal algorithms and lower bounds for clustering and Minimum Spanning Tree (MST) in this model. For k-centers, k-median, and k-means, we give constant factor approximation algorithms with only $\tilde{O}(k)$ strong oracle point queries, and prove that $\Omega(k)$ queries are required for any bounded approximation. For edge queries, our upper and lower bounds are both $\tilde{\Theta}(k^2)$. Surprisingly, for the MST problem we give a $O(\sqrt{\log n})$ approximation algorithm using no strong oracle queries at all, and we prove a matching $\Omega(\sqrt{\log n})$ lower bound which holds even if $\tilde{\Omega}(n)$ strong oracle point queries are allowed. Furthermore, we empirically evaluate our algorithms, and show that their quality is comparable to that of the baseline algorithms that are given all true distances, but while querying the strong oracle on only a small fraction ($< 1\%$) of points.

Keywords: Clustering, Minimum Spanning Tree, Noisy Distances

1. Introduction

Large-scale similarity models are ubiquitous in modern machine learning, where they are used to generate real-valued distances for non-metric data, such as images, text, and videos. A popular example is embedding models (Mikolov et al., 2013; Van der Maaten and Hinton, 2008; He et al., 2016; Devlin et al., 2018), which transform a data point x into a point $f(x)$ in a metric space $(\mathcal{X}, d)$, such that the similarity between x, y can be inferred by the distance $d(f(x), f(y))$. However, as the scale and quality of these models grow, so too increases the resources required to run them. Thus, a common component of many ML pipelines is to additionally employ an efficient but less precise similarity model to reduce the number of expensive distance comparisons made with the more accurate model Kusner et al. (2015); John et al. (2020). Common examples of such "weak" secondary similarity models include hand-crafted models based on simple features (location, timestamp, bitrate, etc.), lightweight neural network, models trained on cheap but sometimes inaccurate data John et al. (2020), meta-data obtained in video transcoding (John et al., 2020; Ringis et al., 2021), previously computed similarities from historical data Mitzenmacher and Vassilvitskii (2022), and the retrieve-then-rerank architecture for recommendation systems Liu et al. (2022), text retrieval Zhang et al. (2022), question-answering Barz and Sonntag (2019) and vision-applications Zhong et al. (2017).

Understanding the complexity of computational tasks in the presence of noisy or imprecise oracles is a fundamental problem dating back multiple decades Feige et al. (1994), and many problems such as clustering, sorting, and nearest neighbor search have been intensively studied therein (Braverman and Mossel, 2008, 2009; Mazumdar and Saha, 2017; Green Larsen et al., 2020; Mason et al., 2019). However, despite the popularity of combining two oracles in practice, the majority of this line of work considers only a single imprecise oracle, whereas much less work has been done to understand the complexity of tasks using *both* a noisy (weak) oracle, and an exact (strong) oracle. In this paper, we initiate a formal study of this setting for metric optimization problems.

Specifically, we introduce the *Weak-Strong Oracle Model*: here, we are given a metric space $(\mathcal{X}, d)$ of $|\mathcal{X}| = n$ points, where $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the underlying metric, representing the output of an expensive but accurate similarity model, as well as a corruption probability $\delta \in (0, 1/2)$. The metric d is not known to the algorithm a priori, but can be accessed through two types of queries: *strong* and *weak oracle* queries. For the strong oracle, we consider two types of queries: edge queries and point queries. The queries are as follows:

- **Weak oracle queries ($\mathsf{WO}(x, y)$):** given $(x, y) \in \mathcal{X}^2$, the weak oracle returns a value $\tilde{d}(x, y)$ such that: with probability $1 - \delta$ we have $\tilde{d}(x, y) = d(x, y)$, and otherwise, with probability δ, the value $\tilde{d}(x, y)$ is set arbitrarily.[1] The randomness is *independent* across different pairs (x, y), and drawn *exactly once* (i.e., repeated queries to $\tilde{d}(x, y)$ will yield the same result).

- **Strong oracle (point) queries ($\mathsf{SO}(x)$):** given a point $x \in \mathcal{X}$, a strong point oracle returns a symbolic value $\mathsf{SO}(x)$. The value $\mathsf{SO}(x)$ gives no information on its own. However, given any two values $\mathsf{SO}(x), \mathsf{SO}(y)$, the algorithm can compute the true distance $d(x, y)$.

- **Strong oracle (edge) queries ($\mathsf{SO}(x, y)$):** given $x, y \in \mathcal{X}$, a strong edge oracle returns the true distance $\mathsf{SO}(x, y) = d(x, y)$.

The weak oracle distances $\tilde{d}$ capture a cheap but less precise distance model, whereas the strong oracle is considered to be significantly more expensive. As a result, our goal is to produce a high-quality solution to an optimization problem (e.g. clustering) for the underlying metric $(\mathcal{X}, d)$ while minimizing the number of queries made to the strong oracle. We even allow the corruptions that occur to the weak oracle to be *adversarial* (see Section 2

1. A pair (x, y) such that the weak oracle distance $\tilde{d}(x, y)$ can be set arbitrarily is called *corrupted*.

for precise model definitions); this captures a very general class of "imprecise weak oracles", allowing them to produce arbitrary bad distances with some probability.

The primary motivation for strong oracle *point* queries is expensive embedding models: in this case, $\mathsf{SO}(x)$ represents the embedding in $(\mathcal{X}, d)$ (e.g., $\mathsf{SO}(x)$ could be a vector in $\mathbb{R}^d$). Clearly, given two such embeddings $\mathsf{SO}(x), \mathsf{SO}(y)$ one can easily compute their distance $d(\mathsf{SO}(x), \mathsf{SO}(y))$. Conversely, the strong oracle edge query model is natural in settings where the expensive oracle does not produce embeddings, but instead directly computes pair-wise similarities, such as cross-attention models (Brown et al., 2020; Thoppilan et al., 2022).

Thus, depending on the context, it may make sense to allow only one of the two types of strong oracle queries. Therefore, we consider *two distinct models*: **(1)** where only strong oracle point queries are allowed, and **(2)** where only strong oracle edge queries are allowed.[2] In this paper, we will give algorithms and lower bounds for both models.

We focus on clustering, which is one of the most fundamental unsupervised learning tasks, and the classic metric minimum spanning tree (MST) problem, which has applications to network design and hierarchical clustering. Both tasks have been studied extensively in the literature on noisy oracles (Ashtiani et al., 2016; Mazumdar and Saha, 2017; Ailon et al., 2017; Ergun et al., 2022; Nguyen et al., 2023; Silwal et al., 2023). However, given the strong type of inaccuracies allowed by our weak oracle, it is not clear whether we can solve these foundational tasks without querying the strong oracle for essentially all the distances. Specifically, we pose the following question:

Is it possible to solve metric optimization tasks, like clustering and MST, in the Weak-Strong Oracle Model while making fewer that $\Omega(n)$ strong oracle point queries (or $\Omega(n^2)$ edge queries)?

1.1. Contributions

Our main contribution is to answer the above question in the affirmative. Specifically, we design constant factor approximation algorithms for k-centers, k-means, k-medians with $\tilde{O}(k)$ point queries to the strong oracle.[3] For MST, we design an algorithm that achieves a $O(\sqrt{\log n})$ approximation without *any* strong oracle queries. For both problems, we prove matching or nearly matching lower bounds, demonstrating the optimality of our algorithms. Our results for k-clustering hold for any corruption probability $\delta < \frac{1}{2}$ bounded away from $\frac{1}{2}$ by a constant, and for MST our results hold for any $\delta < 1$ bounded away from 1. For simplicity, in the statement of our results we fix the corruption probability to be $\delta = 1/3$.

Clustering. We begin with our results for k-clustering. Formally, the problem is as follows: we are given query access (via the weak and strong oracle as defined above) to distances in a metric space $(\mathcal{X}, d)$, and our goal is to output a set of k centers $c_1, \ldots, c_k \in \mathcal{X}$, as well as a mapping $\mathcal{C} : \mathcal{X} \to \{c_i\}_{i=1}^k$, that minimizes the k-clustering cost with respect to the original metric $(\mathcal{X}, d)$. For the k-centers problem, the objective is to minimize $\max_{p \in \mathcal{X}} d(p, \mathcal{C}(p))$. For k-means and k-medians, the goal is to minimize $\sum_{p \in \mathcal{X}} d^q(p, \mathcal{C}(p))$, where $q = 1$ for k-median

2. Note that the two types of strong oracle queries are closely related. For instance, any algorithm that makes q strong oracle point queries can be easily simulated by q^2 strong oracle edge queries.

3. Throughout, we write $\tilde{O}$ to suppress $\log n$ factors.

	# of Strong Point Queries	# of Strong Edge Queries	Approximation	Note
k-clustering	$\tilde{O}(k)$	$\tilde{O}(k^2)$	$O(1)$	Theorems 1 & 2
	$\Omega(k)$	$\Omega(k^2)$	–	Theorem 29 Holds for any approx.
Metric MST	–	–	$O(\sqrt{\log n})$	Theorem 4
	$\tilde{\Omega}(n)$	$\tilde{\Omega}(n)$	$\Omega(\sqrt{\log n})$	Theorem 30
Non-metric MST	$O(n)$	$O(n^2)$	Exact	Trivial
	$\Omega(n)$	$\Omega(n)$	$\Omega(\log n)$	Theorem 32

Table 1: Summary of our results. The top row of each problem contains the upper bound and the bottom row contains the lower bound. Our Metric-MST algorithm from Theorem 4 only uses the weak oracle, and therefore makes no strong oracle queries.

and $q = 2$ for k-means. We emphasize that our clustering algorithms require only *one* of either strong oracle edge queries or strong oracle point queries, and not both.

Theorem 1 (k-Centers Upper Bound) *For any $\varepsilon > 0$ and metric space $(\mathcal{X}, d)$, there is an algorithm in the weak-strong oracle model that, with probability $1 - \frac{1}{\mathrm{poly}(n)}$, obtains a $(14+\varepsilon)$-approximation to k-centers using either $O(k \log^2 n \cdot \log(\frac{\log n}{\varepsilon}))$ strong oracle point queries, or $O(k^2 \log^4 n \cdot \log^2(\frac{\log n}{\varepsilon}))$ strong oracle edge queries. Moreover, the algorithm runs in time $\tilde{O}(nk \log(\frac{1}{\varepsilon}))$.*

Theorem 2 (k-Means and k-Medians Upper Bound) *There exists an algorithm in the weak-strong oracle model that, with probability $1 - \frac{1}{\mathrm{poly}(n)}$, computes an $O(1)$-approximate solution to k-means and k-medians using $O(k \log^2 n)$ strong oracle point queries, or $O(k^2 \log^4 n)$ edge queries. The algorithms runs in time $\tilde{O}(nk + k^3)$.*

Despite the similar query-complexities, the clustering algorithms from Theorems 1 and 2 require very different techniques. Moreover, since it is NP-Hard to give better than a 2 approximation to any of the above clustering tasks (Hsu and Nemhauser, 1979; Cohen-Addad and Karthik, 2019; Cohen-Addad and Lee, 2022), our algorithm's approximations are optimal up to a constant (among polynomial time algorithms).

Next, we demonstrate that the strong-oracle query complexity of *all* our k-clustering algorithms are tight in a *strong* sense. Specifically, we demonstrate that $\Omega(k)$-strong oracle queries are necessary for any algorithm that achieves *any* bounded approximation for k-clustering tasks. This settles the query complexity of this problem up to $\log n$-factors. The proofs of our lower bounds, including Theorems 3, 5, and 6, are deferred to Appendix F.

Theorem 3 (Clustering Lower Bound) *Fix any $c \in \mathbb{R}^+$, k larger than some constant, and corruption probability $\delta = 1/3$. Then any algorithm which obtains a multiplicative c-approximation, with probability at least $1/2$, to either k-centers, k-means, or k-medians, must make at least $\Omega(k)$ strong oracle point queries, or $\Omega(k^2)$ strong oracle edge queries.*

Minimum Spanning Tree. In the classic metric MST problem, the goal is to produce a spanning tree T over the points in $\mathcal{X}$ which minimizes the weight of the tree with respect to the metric $(\mathcal{X}, d)$: namely $w(T) = \sum_{\text{edge } (x,y) \in T} d(x,y)$. We consider the problem in two settings, corresponding to whether or not the weak oracle distances $\tilde{d} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are themselves a metric over $\mathcal{X}$. We refer to the case where $(\mathcal{X}, \tilde{d})$ is restricted to being a metric as the *metric-weak oracle* setting. This setting is especially motivated by weak oracles which are themselves embedding models, such as lighter-weight embeddings or pre-computed embeddings trained on stale or possibly inaccurate data. We demonstrate that, perhaps surprisingly, given a metric weak oracle, we can obtain a good approximation to the optimal MST without resorting to the strong oracle at all.

Theorem 4 *There is an algorithm that, given only access to the corrupted distances $\tilde{d}$ produced by a metric weak oracle (namely, $(\mathcal{X}, \tilde{d})$ is metric) with corruption probability δ such that $1 - \delta \geq c$ for some constant $c > 0$, produces a tree $\hat{T}$ such that $\mathbb{E}[w(\hat{T})] \leq O(\sqrt{\log n}) \cdot \min_T w(T)$. The algorithm runs in time $O(n^2)$.*

A natural question that arises following our MST algorithm is whether a constant approximation is possible, perhaps by allowing for a small number of strong oracle queries as well. We demonstrate, however, that this is impossible in a strong sense: any algorithm that achieves a better than $O(\sqrt{\log n})$ approximation must essentially query the strong oracle for all the distances in $\mathcal{X}$.

Theorem 5 *There exists a constant c such that any algorithm which outputs a spanning tree T such that $\mathbb{E}[w(T)] \leq c\sqrt{\log n} \cdot \min_{T'} w(T')$ must make at least $\Omega(n/\sqrt{\log n})$ strong oracle point queries. Moreover, this holds even when the weak-oracle distances $\tilde{d} : \mathcal{X}^2 \to \mathbb{R}$ are restricted to being a metric, and when the corruption probability is $\delta = 1/3$.*

Theorems 4 and 5 prove tight bounds for the approximation of MST in the metric weak oracle setting. A final question is whether a $O(\sqrt{\log n})$ approximation is possible without the metric restriction on $\tilde{d}$. We demonstrate that this too is impossible, by proving a $\Omega(\log n)$ approximation lower bound in the general case. Combine with our upper bound in Theorem 4, this proves a strong separation between the metric and non-metric weak oracle models.

Theorem 6 *There exists a constant c such that any algorithm which outputs a spanning tree T of $(\mathcal{X}, d)$ such that $\mathbb{E}[w(T)] \leq c \log n \cdot \min_{T'} w(T')$ in the weak-strong oracle model (with corruption probability $\delta = 1/3$), must make at least $\Omega(n)$ queries to the strong oracle.*

We conjecture that the lower bound from Theorem 6 is tight, and that an algorithm exists with *no* strong oracle queries and a $O(\log n)$ approximation. We leave it as an open question for future work to determine the exact query complexity of non-metric MST in the weak-strong oracle model.

Experiments. We empirically evaluate the performance of our algorithms on both synthetic and real-world datasets. For the synthetic data experiments, we use the extensively studied Stochastic Block Model (Holland et al., 1983; Dyer and Frieze, 1989; Decelle et al., 2011; Abbe et al., 2015; Abbe and Sandon, 2015; Hajek et al., 2016; Mossel et al., 2015), which has a natural interpretation as clustering with faulty oracles Mazumdar and Saha (2017). For the real-world dataset, we run experiments to cluster embeddings of the MNIST dataset Deng (2012); specifically, we consider both the SVD and t-SNE embeddings Van der Maaten and Hinton (2008). Our experiments demonstrate that our algorithms achieve clustering costs that are competitive with standard benchmark algorithms that have access to strong oracle queries on the *entire dataset*, while our algorithms only make strong oracle queries on a small fraction of the points (i.e. 1-2% of the points). Furthermore, we show that benchmark algorithms with *no* strong oracle queries produce significantly worse clusterings than our algorithms, demonstrating the necessity of exploiting the strong oracle. Our experimental results are deferred to Appendix G.

1.2. Technical Overview

Perhaps the core challenge in the weak-strong oracle model is that it is often difficult, and sometimes even impossible, to detect when a weak-oracle distance $\tilde{d}(x,y)$ is the true distance $d(x,y)$, or has been arbitrarily corrupted. However, to solve k-clustering or MST using $o(n)$ strong oracle point queries, we must necessarily rely on the weak-oracle for the majority of points. For instance, suppose we were handed the optimal centers $c_1^*, \dots, c_k^* \subset \mathcal{X}$ to a k-clustering problem, and asked to find a good assignment $\mathcal{C} : \mathcal{X} \to \{c_i^*\}_{i=1}^k$. Clearly we cannot assign a point x to $\min_i \tilde{d}(x, c_i^*)$, since a corruption of just one of these distances could lead to an arbitrarily bad solution.

Our first key observation is the following. Suppose we had a set of points U that we knew were close to some c_i^*, e.g. $U \subset \mathcal{B}(c_i^*, r)$ for some small radius r, where $\mathcal{B}(c_i^*, r) = \{y \in \mathcal{X} \mid d(c_i^*, y) \leq r\}$. For a given point $x \in \mathcal{X}$, while some of the distances $\{\tilde{d}(x,u)\}_{u \in U}$ may be corrupted, it is unlikely that all will be. Specifically, if the corruption probability is $\delta = 1/3$, then for large enough U, with high probability less than half of the distances will be corrupted. In particular, it follows that the *median* distance $\{\tilde{d}(x,u)\}_{u \in U}$ will be between $\min_{u \in U} d(x,u)$ and $\max_{u \in U} d(x,u)$. This allows us to prove the following Lemma.

Lemma 7 (Lemma 16 in Appendix B) *Fix any point $u \in \mathcal{X}$, radius $r \geq 0$, and $U \subset \mathcal{B}(u,r)$ with $|U| = \Omega(\log n)$. For any $x \in \mathcal{X}$, define $d_U^{est}(x,u) = \mathsf{Median}\{\tilde{d}(x,y) \mid y \in U\}$. Then with high probability, for all $x \in \mathcal{X}$ we have $|d_U^{est}(x,u) - d(x,u)| \leq r$.*

In other words, Lemma 7 allows us to use the median weak-oracle distance between x and $u \in U$ as a *proxy* for the distance from x to any point in U, with error depending on the diameter of U. This fact will be the basic building block for our k-clustering algorithms, which we now describe. In the following discussion, assume that we have a value r such that $2\mathsf{OPT} \leq r \leq (2+\varepsilon)\mathsf{OPT}$ (we will later find r by guessing it in powers of $(1+\varepsilon)$, and verify when the guess is incorrect).

k-Centers Algorithm. Recall that for k-center clustering, the goal is to minimize the objective $\max_{p \in \mathcal{X}} d(p, \mathcal{C}(p))$. By Lemma 7, to obtain a constant factor approximation it will suffice to identify sets of points $U_1, \dots, U_k \subset \mathcal{X}$ such that $|U_i| = \Omega(\log n)$ and $U_i \subset \mathcal{B}(c_i^*, 3r)$ for each i, where $\{c_i^*\}_{i=1}^k$ is the optimal clustering. Given such sets U_i, we can pick an arbitrary point $c_i \in U_i$ as our chosen center, and assign each point $x \in \mathcal{X}$ to the center c_i that minimizes $d_{U_i}^{est}(x, c_i)$. Since $d(c_i, c_i^*) \leq 3r = O(\mathsf{OPT}_{\text{k-center}})$, by Lemma 16 the result gives a constant approximation.

Our algorithm will (roughly) attempt to recover such sets $U_1, \ldots, U_k$ via a recursive procedure. Specifically, at each step, we will try to assign a constant fraction of the points to cluster centers within distance at most $O(r)$, and recurse on the remaining un-clustered points. Let $C_1^*, \ldots, C_k^* \subset \mathcal{X}$ be the optimal clustering. We first sample a set S of $O(k \log n)$ points uniformly from $\mathcal{X}$ and queries the strong point oracle on S; by standard concentration bounds, for any optimal cluster C_i^* containing at least a $\frac{1}{10k}$-fraction of the points, S will contain at least $\Omega(\log n)$ points from C_i^* with high probability. We call such a cluster C_i^* *heavy*. Thus it will suffice to cluster the points in each heavy cluster C_i^*, remove the clustered points, and recurse on the remaining points. Since at least a $\frac{9}{10}$ fraction of points are in heavy clusters, the recursion will complete after $O(\log n)$ iterations.

The main challenge is to identify the points that belong to a heavy cluster C_i^*. To do this, using a ball-carving procedure, we greedily partition the points in S into k' clusters $B_1, \ldots, B_{k'} \subset S$, centered at $c_1, \ldots, c_{k'} \in S$ respectively, each with radius at most $3r$. We prove that either $k' \leq k$, or the optimal k-centers solution is larger than r, which implies that we guessed r incorrectly. Observe that we can compute this clustering since we queried the strong oracle on S, and therefore know all true pair-wise distances between points in S. Since each B_i has radius $3r > 3\mathsf{OPT}$, each B_i fully contains at least one optimal cluster C_j^*. Thus $B_i \subset \mathcal{B}(c_j^*, 6r)$ for some $j \in [k]$, and moreover we know that every heavy cluster C_j^* will be contained inside such a ball B_i with $|B_i| = \Omega(\log n)$. Using the estimator from Lemma 16, this will allows us to assign all points in heavy clusters C_j^* to (one or more) balls B_i that cover them at distance $O(r)$, thereby removing a $\frac{9}{10}$ fraction of points on that round. The full procedure and proofs for our k-centers algorithm are deferred to Appendix B.

k-Means and k-Medians Algorithm. Our algorithm for k-means and k-median is a more nuanced procedure; let us focus on k-medians for simplicity. Our goal will be to construct a *coreset* S for the k-medians problem, which (roughly) is a set $S \subset \mathcal{X}$ and a mapping $w : \mathcal{X} \to S$ that has the property that the cost of any k-medians solution on S (weighted by the mapping w) approximates the cost on the original set $\mathcal{X}$. Techniques for building such coresets for k-means and medians are well understood in the setting that all distances are available to us, however such methods are brittle to corruptions in the underlying metric. One classic such algorithm is based on the sequential Meyerson's sketch Charikar et al. (2003); Meyerson (2001), which incrementally builds up a coreset S by, for each point $x \in \mathcal{X}$ in some order, adding x to S with probability proportional to its distance $d(x, S)$ from S. Our high-level approach will be to approximately simulate this algorithm.

Unfortunately, unlike in the case of k-centers, even when $|S| = \Omega(\log n)$ we can no longer simply use Lemma 16 to approximate $d(x, S)$, since the diameter of S, and therefore the error from Lemma 16, can be much larger than the cost which x should pay to move to its center in the optimal solution. Thus, we need a more fine-grained corruption-robust distance estimator. Our main insight is that we can design such an estimator, which we call the Heavy-Ball distance (Definition 8), by patching together the estimator Lemma 16 over multiple "small" local subsets $U \subset S$. Specifically, the heavy ball distance approximates $d(x, S)$ via $\min_{y \in S} d^{\text{est}}_{U(y)}(x, y)$, where $U(y) = \mathcal{B}(y, r) \cap S$ is the ball around y with the smallest radius r such that $|U(y)| = \Omega(\log n)$. Our main technical contribution is analyzing the error introduced into the Meyerson's sketch when using the Heavy-Ball distance. See Section 3 for the full algorithm and further detail, and Appendix C for the complete proofs.

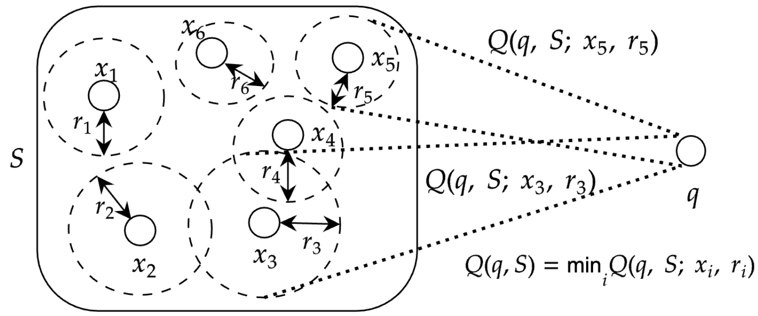


Figure 1: An illustration of heavy-ball nearest distance. Each circle formed by the dashed lines is the ball with the minimum r_i for x_i to have $O(\log n)$ points within. As such, the value of $Q(q, S)$ becomes $\min_i Q(q, S; x_i, r_i)$ in the figure.

MST Algorithms Perhaps our most surprising result is that, in the metric weak-oracle setting, the Minimum Spanning Tree problem can be approximated to a bounded $O(\sqrt{\log n})$ factor without *any* strong oracle queries, and moreover that this approximation is optimal even if we are allowed to query the strong oracle for nearly *all* the distances. Our algorithm itself is simple: it computes an optimal MST $T^*_{\tilde{d}}$ of the corrupted metric $(\mathcal{X}, \tilde{d})$, transforms $T^*_{\tilde{d}}$ into a bounded degree tree $\hat{T}$ (via a relatively simply procedure, see Algorithm 5), and then outputs $\hat{T}$. In other words, simply returning (roughly) the corrupted MST suffices in the metric weak-oracle setting. However, the analysis of this algorithm is fairly involved. We present a simplified version of the analysis here which achieves a $O(\log n)$ approximation, and give the more nuanced construction behind the $O(\sqrt{\log n})$ approximation in Section 4.

The main challenge will be to show that for *every* spanning tree T of $\mathcal{X}$ the cost $w(T)$ of T under the metric d is not much larger than its cost $\tilde{w}(T)$ the under the metric $\tilde{d}$ (the opposite inequality will be easier, since we will only need to do it for a single tree). We first (in the analysis only) partition $\mathcal{X}$ into metric balls $\mathcal{B}_1, \ldots, \mathcal{B}_k$, centered at points $x_1, \ldots, x_k$, with the smallest possible radii $r_1, \ldots, r_k$ such that $|\mathcal{B}_i| = \Theta(\log n)$ for each i (one can do this greedily by picking an arbitrary $x \in \mathcal{X}$ and covering its nearest $O(\log n)$ points). The key property of the construction we prove is that optimal MST cost is at least $\frac{1}{2}\sum_{i=1}^k r_i$.

To bound the cost $w(T)$, we will use a charging scheme to bound the distances $d(x, y)$ for each edge $(x, y) \in T$. Now for any pair of points $x \in \mathcal{B}_i, y \in \mathcal{B}_j$, since each of $\mathcal{B}_i, \mathcal{B}_j$ has size $\Omega(\log n)$, with high probability there will be at least one other point $z \in \mathcal{B}_i$ such that both $d(x, z)$ and $d(z, y)$ are uncorrupted (so $d(x, z) = \tilde{d}(x, z)$ and $d(z, y) = \tilde{d}(z, y)$). If this is the case, then even though $\tilde{d}(x, y)$ could have been corrupted, it cannot be larger than $d(x, z) + d(z, y)$, since otherwise $\tilde{d}(x, y) > \tilde{d}(x, z) + \tilde{d}(z, y)$, thus violating triangle inequality for $\tilde{d}$. Moreover, since $z \in \mathcal{B}_i$, it follows that $\tilde{d}(x, z) \le O(r_i) + d(x, y)$. To analyze the cost $w(T)$, we can charge the extra cost $O(r_i)$ of the edge (x, y) to the ball $\mathcal{B}_i$; since $|\mathcal{B}_i| = \Theta(\log n)$, each ball is charged at most $O(\log n)$ times, and since $\mathsf{OPT} \ge \frac{1}{2}\sum_{i=1}^k r_i$

it follows that the total extra "charged" cost is at most $O(\log n \cdot \mathsf{OPT})$ as desired. The improvement to a $O(\sqrt{\log n})$ approximation requires further nuances, see Section 4 for details.

Lower Bounds For k-clustering, our hard instance is as follows: we pick an arbitrary set S of $\frac{3}{2}(k-1)$ points (assume for simplicity that k is odd), and let O be the remaining points. Then we sample a random subset $N \subset S$ of $k-1$ points, and generate a random perfect matching M over N. We set the $d(x,y)=1$ between matched points $(x,y) \in M$, and also set $d(a,b)=1$ for all $a,b \in O$; all other distances in d are set to an arbitrarily large value $R \gg n$. This creates a clear set of k-clusters (for any k-clustering objective), where each of the $(k-1)/2$ matched pairs in M are clustered together, the $(k-1)/2$ unmatched points in $S \setminus N$ are each put in their own cluster, and all of O is put in a single cluster; any other clustering will incur an arbitrarily large cost. However, for large enough k it is likely that at least one matched pairs $(x,y) \in M$ is corrupted. The weak oracle can then set $\tilde{d}(x,y)=R$, thereby making x,y indistinguishable from all other points in $S \setminus N$. We will show that one must make $\Omega(|S \setminus N|) = \Omega(k)$ strong oracle point queries to recover the matched pair (x,y).

For the general (non-metric) MST lower bound, the hard instance is as follows. We randomly split $\mathcal{X}$ into n/k clusters $C_1, \ldots, C_{n/k}$, each with k points, where $k = \varepsilon \log n$ for small enough constant ε. We set $d(x,y)=1$ if x,y are in the same cluster, and $d(x,y)=k$ otherwise. Clearly, the optimal MST first connects together points within a cluster C_i arbitrarily, and then uses one edge per cluster to connect together the n/k clusters. Now observe that for any pair $x \in C_i, y \in C_j$ in different clusters, with probability $\delta^{2k-2} = \Omega(\frac{1}{n^\varepsilon})$ all distances $\{\tilde{d}(x,w)\}_{w \in C_i} \cup \{\tilde{d}(y,z)\}_{z \in C_j}$ are corrupted. Call such a pair "matchable". If (x,y) is matchable, the weak oracle can set $\tilde{d}(x,z)=1$ for all $z \in C_j$ and $\tilde{d}(y,w)=1$ for all $w \in C_i$, making it impossible to tell whether $x \in C_i, y \in C_j$ or $y \in C_i, x \in C_j$ (without the strong oracle). To prove a $\Omega(k)$ approximation lower bound, the algorithm must make mistakes on $\Omega(n)$ points. Since a pair (x,y) is matchable with probability $\Omega(\frac{1}{n^\varepsilon})$, and this is independent across pairs, using tools from random graph theory we show that there is a proper matching M of size $\Omega(n) \subset \mathcal{X} \times \mathcal{X}$ between matchable pairs. We set $\tilde{d}(a,b)=1$ for all $a,b \in M$, making it impossible to correctly identify most points' clusters without $\tilde{\Omega}(n)$ strong oracle queries. Our construction for metric weak-oracle MST is similar, except for a pair $x \in C_i, y \in C_j$ to be matchable we need to corrupt *all* $|C_i| \cdot |C_j|$ distances between C_i and C_j, otherwise our corrupted distances $\tilde{d}$ would violate the triangle inequality, thus failing to be a metric. Therefore, we can only afford to set $k = \sqrt{\varepsilon \log n}$, thereby obtaining a $\Omega(\sqrt{\log n})$ lower bound via a similar approach as the non-metric case.

1.3. Other Related Work

The weak-strong oracle model introduced in this paper is closely related to both *active learning* and *clustering under budget constraints*, which limit the number of pair-wise comparisons. For the two oracle setting, active learning with both weak and strong labelers Zhang and Chaudhuri (2015); Younesian et al. (2020), as well as active learning with diverse labelers Huang et al. (2017) have been studied. In the budget constrained clustering case, a line of work considered spectral clustering on partially sampled matrices Fetaya et al. (2015); Shamir and Tishby (2011); Wauthier et al. (2012), and García-Soriano et al. (2020) devise correlation clustering algorithms with approximation depending on the query budget. Two other closely related lines of work are *clustering with noisy oracles* and *algorithms with predictions* (see

Mitzenmacher and Vassilvitskii (2022) for a survey of the latter[4]). Many tasks, including correlation clustering and signed edge prediction Mazumdar and Saha (2017); Mitzenmacher and Tsourakakis (2016); Green Larsen et al. (2020), k-clustering Ashtiani et al. (2016); Ailon et al. (2017); Nguyen et al. (2023); Addanki et al. (2021); Ergun et al. (2022), and MST Erlebach et al. (2022); Berg et al. (2023), have been studied.

The key difference between all the aforementioned settings and ours is that they are given free access to the true similarities (i.e. $(\mathcal{X}, d)$ for us), and their noisy queries provide access to the *optimal* clustering (or ground truth labels); for instance, their oracles can be asked queries like "should x and y be clustered together"? Comparatively, in our setting the strong oracle simply provides non-noisy access to the input distances. For such oracles, perhaps the most closely related work is the recent paper Silwal et al. (2023), which studies *correlation clustering* with weak and strong oracles akin to ours. However, their model is limited to correlation clustering where the input is a graph with binary labels, whereas our model is based in a metric space, and thus captures metric optimization problems. For the setting where we only employ a weak oracle (such as our MST algorithm), perhaps the most closely related work is Mason et al. (2019), which studies finding nearest neighbors when distances are corrupted by Gaussian noise, which is an incomparable setting to ours.

2. Preliminaries

A full instance of the weak-strong oracle model is specified by the triple $(\mathcal{X}, d, \tilde{d})$, where $\tilde{d} : \mathcal{X}^2 \to \mathbb{R}$ are the distances returned by the weak oracle. We write $\textsc{Corrupt} \subseteq \binom{n}{2}$ to denote the set of "corrupted" distances (pairs (x, y) where $\tilde{d}(x, y) \neq d(x, y)$). We allow the values of $\tilde{d}(x, y)$ for $(x, y) \in \textsc{Corrupt}$ can be chosen arbitrarily and by an adversary who knows the full metric $(\mathcal{X}, d)$ as well as the set $\textsc{Corrupt}$. We write $\Delta \geq 1$ to denote the *aspect ratio* of the original metric space $(\mathcal{X}, d)$. Without loss of generality (via scaling), we can assume that $1 \leq d(x, y) \leq \Delta$ for all $x, y \in \mathcal{X}$. Note that this bound only applies to the strong oracle – the weak oracle distances $\widetilde{d}$ can of course be arbitrarily larger than Δ or smaller than 1. Throughout, we will assume that the aspect ratio is polynomially bounded, namely that $\Delta \leq n^c$ for any arbitrarily large constant $c \geq 0$, which is a common assumption in the literature. We discuss the generalization to arbitrary aspect ratio in Appendix A.2.

Our algorithms for k-centers, k-means, and k-medians will only need to use strong oracle point queries; since any algorithm that makes $\tilde{O}(k)$ point queries can be transformed into an algorithm that makes at most $\tilde{O}(k^2)$ edge queries (simply by querying all stances between the set $S \subset \mathcal{X}$ of point-queries), the edge query complexity follows as a corollary. Thus, in what follows, "strong oracle query" refers to strong oracle point queries, and edge queries will be explicitly specified as such. For simplicity, we present our clustering algorithms for the case of $\delta = 1/3$, and describe in Appendix D the generalization to any $\delta \in (0, 1/2)$.

Notation. For a set $S \subset \mathcal{X}$, we write $d(u, S) = \min_{y \in S} d(x, y)$ and $\widetilde{d}(u, S) = \min_{y \in S} \widetilde{d}(x, y)$. For a metric space $(\mathcal{X}, d)$, a point $x \in \mathcal{X}$, and a radius $r > 0$, we write $\mathcal{B}_d(x, r) = \{y \in \mathcal{X} \mid d(x, y) \leq r\}$ to denote the closed metric ball centered at x with radius r under d. When d is the original metric $(\mathcal{X}, d)$, we simply write $\mathcal{B}(x, r) = \mathcal{B}_d(x, r)$. We write $\mathsf{OPT}_{\text{k-center}}(d)$, $\mathsf{OPT}_{\text{k-means}}(d)$, and $\mathsf{OPT}_{\text{k-median}}(d)$ to denote the optimal clustering cost of k-center, k-means and k-median on $\mathcal{X}$ with distance metric d. When the

4. Also see the website https://algorithms-with-predictions.github.io/

problem of study is clear, we also simply use OPT to denote the optimal cost. For the metric minimum spanning tree problem, given a tree $T = (\mathcal{X}, E)$ spanning the points in $\mathcal{X}$, we write $w_d(T) := \sum_{(x,y)\in E} d(x,y)$ to denote the cost of the tree in the metric d, and $w_{\tilde{d}}(T) := \sum_{\text{edge}(x,y)\in T} \tilde{d}(x,y)$ to denote the cost with respects to $\tilde{d}$. For clarity, we will sometimes write $w(T) = w_d(T)$ and $\tilde{w}(T) = w_{\tilde{d}}(T)$.

3. k-Means and k-Median Clustering in the Weak-Strong Oracle Model

We now describe our algorithm for the k-means and k-median clustering (Theorem 2), where the objective is to minimize the sum $\sum_{p\in\mathcal{X}} d^{\ell}(p, \mathcal{C}(p))$, where $\ell = 1$ for k-median and $\ell = 2$ for k-means. The algorithm proceeds by first constructing a *coreset* $S \subset \mathcal{X}$ of at most $O(k \log^2 n)$ points, which contains a set of k centers that are $O(1)$-approximation to OPT. To build the coreset S, we arbitrarily order the points, and for point p (in order), we sample p into S with probability proportional the the distance $d(p, S)$. To approximate $d(p, S)$ in the weak-strong oracle model, we design a *proxy distance* based on the weak oracle. Unfortunately, we cannot simply apply Lemma 7, since the diameter of S may be too large compared to the true distance $d(p, S)$. Thus, we require a more fine-grained estimator, which is as follows:

Definition 8 (Heavy-ball distance) *Fix the metric space $(\mathcal{X}, d)$, let $q \in \mathcal{X}$ be a point and $S \subseteq \mathcal{X} \setminus \{q\}$ be a set of points such that $|S| \geq 100 \log n$. Then for any $x \in S$ and radius $r > 0$, we set $Q(q, S;\ x, r) = \mathsf{Median}\{\widetilde{d}(q, z) \mid z \in S \cap \mathcal{B}(x, r)\} + 6 \cdot r$, and define the* <u>*heavy-ball distance*</u> *as*

$$Q(q, S) = \min_{\substack{x\in S, r>0\\ |\mathcal{B}(x,r)\cap S|\geq 100\log(n)}} Q(q, S;\ x, r).$$

Notice that the value $Q(q, S;\ x, r)$ is the same as our earlier estimator $d^{\text{est}}_{\mathcal{B}(x,r)\cap S}(q, x)$ from Lemma 7, plus an additional $6 \cdot r$ term which is needed to ensure that we always have the lower bound $Q(q, S \mid x, r) \geq d(q, x)$ whenever $|\mathcal{B}(x, r) \cap S| \geq 100 \log(n)$. It will follow that $Q(q, S) \geq d(q, S)$ for all q, S with high probability. Intuitively, the heavy-ball distance approximates $d(q, S) = \min_{x\in S} d(q, x)$ by approximating each $d(q, x)$ via $d^{\text{est}}_U(q, x)$, where U is the smallest radius ball $\mathcal{B}(x, r) \cap S$ around x that has enough points in it for the concentration inequalities from Lemma 7 to apply. Notice, moreover, that $Q(q, S)$ can be computed using only the weak-oracle, so long as the strong (point) oracle has been previously called on all points $x \in S$, since then we have all true pairwise distances $d(x, y)$ for $x, y \in S$.

Unfortunately, while we always have $Q(q, S) \geq d(q, S)$, it will not be possibly to always upper bound $Q(q, S)$, since the nearest neighbor $x \in S$ to q may have very few other points near it, thus the smallest radius r with $|\mathcal{B}(x, r) \cap S| \geq 100 \log n$ may be significantly larger than $d(x, S)$. Our analysis centers around carefully bounding the total errors introduces by *overestimations* produced by $Q(q, S)$ over the complete execution of the algorithm. Our full k-means (and k-median) algorithm is then presented in in Algorithm 1. We prove that, for the correct guess of $\widetilde{\mathsf{OPT}}$, the algorithm samples at most $O(k \log^2 n)$ points (therefore bounding the query complexity), and that this coreset contains a set of k-centers which are a $O(1)$-approximation. We can then run a weighted k-clustering algoithm on the coreset S to find this $O(1)$-approximate solution.

Algorithm 1 **The k-means (and k-median) algorithm**

Input: Estimate of the optimal cost $\widetilde{\mathsf{OPT}}$, arbitrarily ordered points $\mathcal{X} = \{x_1, \cdots, x_n\}$.
Output: Set of centers $C = \{c_1, \cdots, c_m\}$ and assignment $\mathcal{C} : \mathcal{X} \to C$
Init: Set $S = \{x_1\}$, $f = \left(\frac{1}{20} \cdot \frac{\widetilde{\mathsf{OPT}}}{k \log^2 n}\right)$, and initialize weights $w(x_i) = 0$ for $i \in [n]$

for $x_i \in \mathcal{X}$ **do**
  if $|S| < 100 \log n$ **then**
    Add x_i to S, set $w(x_i) = 1$, and query the strong oracle SO on x_i.
  else
    Sample x_i with probability $\min\{1, Q(x_i, S)/f\}$.
    if x_i *is sampled* **then**
      $S \leftarrow S \cup \{x_i\}$, set $w(x_i) = 1$, and query the strong oracle SO on x_i.
    else
      Set $\mathcal{C}(x_i) = x'$, where $x' \in S$ is any point that attains the heavy-ball distance, meaning $Q(x_i, S) = Q(x_i, S; x', r)$ for some r with $|\mathcal{B}(x', r) \cap S| \geq 100 \log n$.
      $w(x') \leftarrow w(x') + 1$.
    end
  end
end

Run any $O(1)$-approximate weighted k-means (resp. k-median) clustering algorithm on S with the weights $\{w(x)\}_{x \in S}$ (e.g. the algorithm of (Mettu and Plaxton, 2004)).

4. Metric Minimum Spanning Tree

In Section 3 we showed that $\tilde{\Theta}(k)$ strong oracle queries are necessary and sufficient for k-clustering tasks. In light of this, a natural question is whether the strong oracle is necessary for all geometric optimization problems. In this section, we demonstrate that, surprisingly, this is not the case for the classic metric minimum spanning tree (MST) problem, so long as the weak distances $\tilde{d} : \mathcal{X}^2 \to \mathbb{R}$ are a metric over $\mathcal{X}$. We refer to this as the *metric-weak oracle model*. We now describe the main ideas behind our main MST algorithm from Theorem 4.

Our algorithm itself is simple: it computes an optimal MST $T^*_{\tilde{d}}$ of the corrupted metric $(\mathcal{X}, \tilde{d})$, transforms $T^*_{\tilde{d}}$ into a bounded degree tree $\hat{T}$ such that $\tilde{w}(\hat{T}) \leq 2\tilde{w}(T^*_{\tilde{d}})$ (a relatively simply procedure, see Algorithm 5), and then outputs $\hat{T}$. The analysis of this algorithm, however, is fairly involved. Let T^*_d be the optimal MST for $(\mathcal{X}, d)$. To prove that $\hat{T}$ is a good approximation, it suffices to show that with good probability we have both:

$$\textbf{(1)}\ w(\hat{T}) \leq \tilde{w}(\hat{T}) + O(\sqrt{\log n})w(T^*_d), \qquad \text{and} \qquad \textbf{(2)}\ \tilde{w}(T^*_d) \leq O(w(T^*_d))$$

Then, using that $\tilde{w}(\hat{T}) \leq 2\tilde{w}(T^*_{\tilde{d}}) \leq 2\tilde{w}(T^*_d)$, we can obtain

$$w(\hat{T}) \leq \tilde{w}(\hat{T}) + O(\sqrt{\log n})w(T^*_d) \leq 2\tilde{w}(T^*_d) + O(\sqrt{\log n})w(T^*_d) \leq O(\sqrt{\log n})w(T^*_d).$$

Which will complete the result, since $w(T^*_d) \leq w(\hat{T})$ by optimality of T^*_d. We begin with the notion of an ℓ-heavy ball, which will be cruicial to the analysis.

Definition 9 (Rephrasing of Definition 23) *Fix any $\ell \geq 0$. For any point $v \in \mathcal{X}$, we define the level-ℓ heavy radius at the point v to be the smallest radius $r = r_v^\ell$ such that $\mathcal{B}_d^\ell(v, r)$ contains 2^ℓ points from $\mathcal{X}$. We define the level-ℓ heavy ball at v to be the ball $\mathcal{B}_d^\ell(v, r_v^\ell)$.*

Next, we design a new ball carving procedure that covers the points in $\mathcal{X}$ with ℓ-heavy balls $\mathcal{B}_1^\ell, \dots, \mathcal{B}_k^\ell$ for any ℓ, where $\mathcal{B}_i^\ell$ has radius r_i^ℓ and is centered at some point x_i. The ball-carving is only used in the analysis, and does not appear in the algorithm. The key property of the ball-carving we prove is that $w(T_d^*) \geq \frac{1}{2} \sum_i r_i^\ell$ for any ℓ. We use this property to charge the extra cost of $w(\hat{T})$ to the radii r_i^ℓ. Now to prove **(1)**, we need to show that for every edge $(x, y) \in \hat{T}$ the distance $d(x, y)$ is not too much larger than $\tilde{d}(x, y)$. We do this in two parts. Define ℓ_1, ℓ_2 so that $2^{\ell_1} = \Theta(\sqrt{\log n})$, and $2^{\ell_2} = \Theta(\log n)$. We say a point p is *ℓ-good* if at least $\Omega(2^\ell)$ distances between x and points in the ball $\mathcal{B}_i^\ell$ containing x are not corrupted, otherwise we say that p is *ℓ-bad*. We then show the following key proposition.

Proposition 27 *With high probability, for any $\ell \geq \ell_1$, for every pair of two ℓ-good points x, y we have $d(x, y) \leq \tilde{d}(x, y) + 4(r_i^\ell + r_j^\ell)$ where (i, j) is such that $x \in \mathcal{B}_i^\ell, y \in \mathcal{B}_j^\ell$.*

Thus, for all edges $(x, y) \in \hat{T}$ where x, y are ℓ_1-good, we have $d(x, y) \leq \tilde{d}(x, y) + 4(r_i^{\ell_1} + r_j^{\ell_1})$, where $x \in \mathcal{B}_i^{\ell_1}$ and $y \in B_j^{\ell_1}$. Since each $\mathcal{B}_i^{\ell_1}$ has at most $2^{\ell_1} = O(\sqrt{\log n})$ points, and since T has bounded degree, each radius $r_i^{\ell_1}$ is charged at most $O(\sqrt{\log n})$ times in the above, so the total extra cost for good edges is $O(\sqrt{\log n}) \sum_i r_i^{\ell_1} \leq O(\sqrt{\log n}) w(T_d^*)$. To handle the ℓ_1-*bad* points, we first prove that *every* point will be at least ℓ_2-good, so we can always use Proposition 27 to charge the extra cost from ℓ_1-bad points to radii $r_j^{\ell_2}$ in the ℓ_2 carving. We then prove that every ball $\mathcal{B}_i^{\ell_2}$ will have at most $\sqrt{\log n}$ ℓ_1-bad points, thus each level-ℓ_2 radius $r_j^{\ell_2}$ is also charged at most $O(\sqrt{\log n})$ times. Our proof of **(2)** proceeds similarly, except that we do not pay the $O(\sqrt{\log n})$ factor since the tree T_d^* is independent of the random corruptions in $\tilde{d}$, thus we can analyze its cost in expectation.

MST Lower Bounds. In addition to our upper bound, we prove a matching $\Omega(\sqrt{\log n})$ approximation lower bound (Theorem 5) for MST in the metric weak-oracle setting. Furthermore, for the general (non-metric) Weak-Strong Oracle model, we prove an even stronger $\Omega(\log n)$ lower bound (Theorem 6), thereby separating the models (see Appendix F).

Acknowledgments

Prathamesh Dharangutte would like to acknowledge NSF support through IIS-2229876 and CCF-2118953. Part of this work was done when Chen Wang was at Rutgers University and was supported in part by NSF CAREER Grant CCF-2047061, a gift from Google Research, and a Fulcrum award from Rutgers Research Council.

References

Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 670–688. IEEE, 2015.

Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on information theory*, 62(1):471–487, 2015.

Raghavendra Addanki, Sainyam Galhotra, and Barna Saha. How to design robust algorithms using noisy comparison oracle. *arXiv preprint arXiv:2105.05782*, 2021.

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 61–72. IEEE Computer Society, 2017. doi: 10.1109/FOCS.2017.15. URL https://doi.org/10.1109/FOCS.2017.15.

Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. *arXiv preprint arXiv:1704.01862*, 2017.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM, 2007. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. *Advances in neural information processing systems*, 29, 2016.

Michael Barz and Daniel Sonntag. Incremental improvement of a question answering system by re-ranking answer candidates using machine learning. In Erik Marchi, Sabato Marco Siniscalchi, Sandro Cumani, Valerio Mario Salerno, and Haizhou Li, editors, *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction - 10th International Workshop on Spoken Dialogue Systems, IWSDS 2019, Syracuse, Sicily, Italy, 24-26 April 2019*, volume 714 of *Lecture Notes in Electrical Engineering*, pages 367–379. Springer, 2019. doi: 10.1007/978-981-15-9323-9_34. URL https://doi.org/10.1007/978-981-15-9323-9_34.

Magnus Berg, Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online minimum spanning trees with weight predictions. *CoRR*, abs/2302.12029, 2023. doi: 10.48550/arXiv.2302.12029. URL https://doi.org/10.48550/arXiv.2302.12029.

Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. SODA '08, page 268–276, USA, 2008. Society for Industrial and Applied Mathematics.

Mark Braverman and Elchanan Mossel. Sorting from noisy information. *arXiv preprint arXiv:0910.1191*, 2009.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 30–39, 2003.

Vincent Cohen-Addad and CS Karthik. Inapproximability of clustering in lp metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539. IEEE, 2019.

Vincent Cohen-Addad and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in lp-metrics. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1493–1530. SIAM, 2022.

Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183. ACM, 2004. doi: 10.1145/1007352.1007386. URL https://doi.org/10.1145/1007352.1007386.

Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Martin E. Dyer and Alan M. Frieze. The solution of some random np-hard problems in polynomial expected time. *Journal of Algorithms*, 10(4):451–489, 1989.

Jon C. Ergun, Zhili Feng, Sandeep Silwal, David Woodruff, and Samson Zhou. Learning-augmented \$k\$-means clustering. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=X8cLTHexYyY.

Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Learning-augmented query policies for minimum spanning tree with uncertainty. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 49:1–49:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.ESA.2022.49. URL https://doi.org/10.4230/LIPIcs.ESA.2022.49.

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

Ethan Fetaya, Ohad Shamir, and Shimon Ullman. Graph approximation and clustering on a budget. In *Artificial Intelligence and Statistics*, pages 241–249. PMLR, 2015.

David García-Soriano, Konstantin Kutzkov, Francesco Bonchi, and Charalampos Tsourakakis. Query-efficient correlation clustering. In *Proceedings of The Web Conference 2020*, pages 1468–1478, 2020.

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

Kasper Green Larsen, Michael Mitzenmacher, and Charalampos Tsourakakis. Clustering with a faulty oracle. In *Proceedings of The Web Conference 2020*, pages 2831–2834, 2020.

Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 359–366. IEEE Computer Society, 2000. doi: 10.1109/SFCS.2000.892124. URL https://doi.org/10.1109/SFCS.2000.892124.

Bruce Hajek, Yihong Wu, and Jiaming Xu. Achieving exact cluster recovery threshold via semidefinite programming: Extensions. *IEEE Transactions on Information Theory*, 62 (10):5918–5937, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Wen-Lian Hsu and George L Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.

Sheng-Jun Huang, Jia-Lve Chen, Xin Mu, and Zhi-Hua Zhou. Cost-effective active learning from diverse labelers. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1879–1885. ijcai.org, 2017. doi: 10.24963/ijcai.2017/261. URL https://doi.org/10.24963/ijcai.2017/261.

Sam John, Akshay Gadde, and Balu Adsumilli. Rate distortion optimization over large scale video corpus with machine learning. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1286–1290, 2020. doi: 10.1109/ICIP40778.2020.9191120.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. Neural re-ranking in multi-stage recommender systems: A review. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 5512–5520. ijcai.org, 2022. doi: 10.24963/ijcai.2022/771. URL https://doi.org/10.24963/ijcai.2022/771.

Blake Mason, Ardhendu Tripathy, and Robert Nowak. Learning nearest neighbor graphs from noisy distance samples. *Advances in Neural Information Processing Systems*, 32, 2019.

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. *Mach. Learn.*, 56(1-3):35–60, 2004. doi: 10.1023/B:MACH.0000033114.18632.e0. URL https://doi.org/10.1023/B:MACH.0000033114.18632.e0.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Michael Mitzenmacher and Charalampos E Tsourakakis. Predicting signed edges with $o(n^{1+o(1)} \log n)$ queries. *arXiv preprint arXiv:1609.00750*, 2016.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.

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.

Thy Dinh Nguyen, Anamay Chaturvedi, and Huy Nguyen. Improved learning-augmented algorithms for k-means and k-medians clustering. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=dCSFiAl_VO3.

Daniel J Ringis, François Pitié, and Anil Kokaram. Near optimal per-clip lagrangian multiplier prediction in hevc. In *2021 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2021.

Ohad Shamir and Naftali Tishby. Spectral clustering on a budget. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 661–669. JMLR Workshop and Conference Proceedings, 2011.

Sandeep Silwal, Sara Ahmadian, Andrew Nystrom, Andrew McCallum, Deepak Ramachandran, and Seyed Mehran Kazemi. Kwikbucks: Correlation clustering with cheap-weak and expensive-strong signals. In *The Eleventh International Conference on Learning Representations*, 2023.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

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

Fabian L Wauthier, Nebojsa Jojic, and Michael I Jordan. Active spectral clustering via iterative uncertainty reduction. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1339–1347, 2012.

Taraneh Younesian, Dick H. J. Epema, and Lydia Y. Chen. Active learning for noisy data streams using weak and strong labelers. *CoRR*, abs/2010.14149, 2020. URL https://arxiv.org/abs/2010.14149.

Chicheng Zhang and Kamalika Chaudhuri. Active learning from weak and strong labelers. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 703–711, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/eba0dc302bcd9a273f8bbb72be3a687b-Abstract.html.

Yanzhao Zhang, Dingkun Long, Guangwei Xu, and Pengjun Xie. HLATR: enhance multi-stage text retrieval with hybrid list aware transformer reranking. *CoRR*, abs/2205.10569, 2022. doi: 10.48550/arXiv.2205.10569. URL https://doi.org/10.48550/arXiv.2205.10569.

Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3652–3661. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.389. URL https://doi.org/10.1109/CVPR.2017.389.

Appendix A. Technical Preliminaries

A.1. Concentration inequalities

We first state several standard concentration inequalities used in the analysis of this paper, beginning with the multiplicative form of the Chernoff bound.

Proposition 10 (Multiplicative Chernoff bound) *Let $X_1, \ldots, X_n$ be n independent random variables with support in $[0, 1]$. Define $X := \sum_{i=1}^{n} X_i$. Then, for every $\gamma > 0$, there are*

$$\Pr\left(X > (1+\gamma)\cdot \mathbb{E}\left[X\right]\right) \leq \exp\left(-\frac{\gamma^2\,\mathbb{E}\left[X\right]}{2+\gamma}\right);$$
$$\Pr\left(X < (1-\gamma)\cdot \mathbb{E}\left[X\right]\right) \leq \exp\left(-\frac{\gamma^2\,\mathbb{E}\left[X\right]}{2}\right).$$

Proposition 10 assumes *independent* random variables. It is known that the Chernoff bound is also applicable in the scenario of *negatively correlated* random variables, defined as follows.

Definition 11 (Negatively Correlated Random Variables) *Random variables $X_1, \ldots, X_n$ are said to be* negatively correlated *if and only if*

$$\mathbb{E}\left[\prod_{i=1}^{n} X_i\right] \leq \prod_{i=1}^{n} \mathbb{E}\left[X_i\right].$$

In particular, if X_i's are independent, we have $\mathbb{E}\left[\prod_{i=1}^{n} X_i\right] = \prod_{i=1}^{n} \mathbb{E}\left[X_i\right]$.

One can easily verify the following sufficient condition to verify the negative correlation between random variables.

Fact A.1 *A sufficient condition for X_i and X_j to be negatively correlated is that conditioning on $X_i = 1$, the probability for $X_j = 1$ does not increase.*

And the concentration in Proposition 10 applies to negatively correlated random variables.

Proposition 12 (Generalized Chernoff) *Let $X_1, \ldots, X_n$ be n negatively correlated random variables supported on $\{0, 1\}$. Then, the concentration inequality in Proposition 10 still holds.*

A.2. Aspect Ratio for Clustering and Guessing $\widetilde{OPT}$

We assumed polynomially-bounded aspect ratio Δ in our work for the simplicity of presentation. However, our results can be easily extended to arbitrarily large Δ by simply increasing the domain of possible guesses for $\widetilde{OPT}$. Specifically, note that only our clustering algorithms depend on the aspect ratio Δ (not our MST algorithm), and this dependency only appears in the universe of possible guesses for an approximation $\widetilde{OPT}$ of the optimal clustering cost (this approximation is needed for k-centers, k-means, and k-median). Specifically, consider the set of $O(\varepsilon^{-1}\log\Delta)$ possible guesses $t = (1+\varepsilon)^0, (1+\varepsilon)^1, \ldots, \Delta$ for the approximate cost

$\widetilde{OPT}$, where we set $\varepsilon = 1$ for k-means and k-medians, and allow for smaller ε for k-centers (as stated in Theorem 1). For each of the clustering problems, it will suffice to find *any* level t such that the guess at t is deemed "too small" by the algorithm (described below), and such that running the algorithm at level $t(1+\varepsilon)$ produces a valid solution.

For k-centers, a level t is deemed as "too small" of a guess if the number of centers produced by that level is larger than k (see Algorithm 2). For k-means and median, the level is "too small" if we sample too many (more than $O(k \log^2 n)$ centers in Algorithm 1. Thus, we can easily run our algorithm on guess of $\widetilde{OPT}$ which are chosen via binary search to find a guess t which is "too small" and such that $t(1+\varepsilon)$ is not too small (and therefore produces a valid solution). This results in an $O(\log \frac{\log \Delta}{\varepsilon})$ factor in the strong oracle query complexity instead of an $O(\log \frac{\log n}{\varepsilon})$ factor. Further problem-specific details are deferred to the proofs of the respective algorithms.

Appendix B. The k-center algorithm

We now give the formal description and analysis of Algorithm 2. For simplicity, we first state the algorithm using monotonically increasing values of R, and we lately describe how this can be sped up via binary searching for a usable value of R.

Algorithm 2 **The k-center algorithm**

Input: Set of points $\mathcal{X}$, number of centers k.
Output: Clustering $\mathcal{C}$ with centers $C = \{c_1, \cdots, c_k\}$ and assignment of each point $x \in \mathcal{X}$
for $R = (1+\varepsilon)^\ell$, $\ell \in [O(\log n)]$ **do**
  Run sample and cover (Algorithm 4) on $\mathcal{X}$ to obtain candidate centers;
  Run greedy ball carving (Algorithm 3) on candidate centers with threshold R;
  If number of centers obtained from the above procedure is $> k$, increase ℓ and repeat.
end

Before describing the procedures and their guarantees, we first introduce some notations. Let $\mathcal{B}_1^* = \mathcal{B}_1^*(c_1^*, \mathsf{OPT}), \cdots, \mathcal{B}_k^* = \mathcal{B}_k^*(c_k^*, \mathsf{OPT})$ centered at $c_1^*, \cdots, c_k^*$ be the clusters corresponding to the optimal k-center solution on $(\mathcal{X}, d)$. We define the "cover" of a point $x \in \mathcal{X}$ as follows.

Definition 13 *For any pair of point $(x, y) \in \mathcal{X}$, we say y is covered by the ball $\mathcal{B}(x, r)$ centered at point x with radius r if $d(x, y) \leq r$ (namely, if $y \in \mathcal{B}(x, r)$).*

We also define the following notion of "heavy ball" that captures the balls with a sufficiently large number of points inside.

Definition 14 *Let $(\mathcal{X}, d)$ be a metric space, and fix any $x \in \mathcal{X}$ and $r > 0$. We say that the ball $\mathcal{B}(x, r)$ is heavy if $|\mathcal{B}(x, r) \cap \mathcal{X}| \geq \frac{n}{10k}$.*

For $\ell \leq k$, denote the corresponding set of *heavy balls* from optimal solution for k-center by

$$\mathcal{B}_{\mathcal{H}}^* = \{\mathcal{B}_1^*(c_1^*, \mathsf{OPT}), \cdots, \mathcal{B}_\ell^*(c_\ell^*, \mathsf{OPT})\}.$$

Observation B.1 *The union of all heavy balls from $\mathcal{B}_H^*$ cover at least $\frac{9n}{10}$ points in $\mathcal{X}$*

Proof Consider the balls in optimal solution of k-center on $(\mathcal{X}, d)$ that are not heavy as per definition 14. There are at most $k-1$ such balls each covering at most $\frac{n}{10k}$ points in $\mathcal{X}$ which amounts to at most $n/10$ points. As a result, heavy balls cover $\frac{9n}{10}$ points in $\mathcal{X}$. ■

We define the "cover" of set of points by a set of balls as follows.

Definition 15 *For a set of points U, we say U is covered by $\mathcal{B}_C = \{\mathcal{B}(c_1, r_1), \cdots, \mathcal{B}(c_\ell, r_\ell)\}$, a set of ℓ balls, if for all $u \in U$ there exists $\mathcal{B}(c_i, r_i) \in \mathcal{B}_C$ such that $\mathcal{B}(c_i, r_i)$ covers u.*

We first describe the procedures "Greedy Ball Carving" and "Sample and Cover" in greater detail, whose guarantees will ultimately be used to complete the proof for theorem 1. We start by assuming knowledge of an estimate $2\mathsf{OPT} \le R \le 2(1+\varepsilon)\mathsf{OPT}$. In the proof for theorem 1 we will guess R in powers of $(1+\varepsilon)$.

We first discuss the procedure greedy ball carving (algorithm 3), which serves as a witness in estimating R. For a set of points with strong oracle queries and some R, the procedure first randomly picks a point as center. Removing all points in the set that are at distance $\le R$ from center, it recurses on the remaining points. We will use the centers generated by algorithm 3 in identification of heavy balls and points within.

Algorithm 3 Greedy Ball Carving

Input: Set of points S, radius R
Output: Set of centers $C = \{c_1, \cdots, c_m\}$ and assignment of $s \in S$ to respective centers.
Init: $C = \{\}$
while *S is not empty* **do**
  Pick arbitrary point $c \in S$
  Treating c as center, assign s to c if $d(c, s) \le R$ for all $s \in S$.
  Add c to C and remove assigned points.
end

Note that greedy ball carving will *always* be used by our algorithm on sets with strong oracle queries and as a result operates with true distances. The following observation helps us to use greedy ball carving as a witness for guessing R, as discussed in section A.2.

Observation B.2 *If OPT is the optimal k-center cost for $(\mathcal{X}, d)$ and $2\mathsf{OPT} \le R$, then algorithm 3 run on any subset of points $S \subseteq \mathcal{X}$, with radius R returns $m \le k$ centers.*

Proof The proof is by contradiction. If the number of centers returned by algorithm 3 is $m > k$, this implies there are $k+1$ points with pairwise distance more than R which contradicts the assumption $2\mathsf{OPT} \le R$. ■

We now describe the recursive procedure sample and cover (Algorithm 4) which at each step aims to cluster points in each heavy balls. We define "complete" balls to be the balls for which we are able to identify $O(\log n)$ points that are covered by respective balls. This is explained more concretely in context of our algorithm in explanation for Step 2, where ultimately the complete balls shall be used for estimating distances.

Algorithm 4 **The recursive sample and cover procedure**

Input: Set of points $\mathcal{X}$, weak distance oracle WO, strong oracle SO, estimate R

Output: Clustering $\mathcal{C}$ with centers $C = \{c_1, \cdots, c_m\}$ and assignment of each point $c \in \mathcal{X}$, and $\mathcal{X}_{\mathsf{SO}}$ the set of points with SO

Init: $C = \{\}$ and $\mathcal{X}_{\mathsf{SO}} = \{\}$

while $\mathcal{X}$ *is not empty* **do**

Sample $|S| = 100k \log n$ and $|T| = 2000k \log n$ points u.a.r. from $\mathcal{X}$

Query SO for $S \cup T$ and set $\mathcal{X} = \mathcal{X} \setminus \{S \cup T\}$.

Step 1. Run algorithm 3 on S with radius R and let the set of centers obtained be C

Check if $|C| > k$

Step 2. Identify complete balls for $c_i \in C$ using T and add $T \cup \{S \setminus C\}$ to $\mathcal{X}_{\mathsf{SO}}$.

for $x \in \mathcal{X} \setminus \{S \cup T\}$ **do**

Step 3. Let $T_i = T \cap \mathcal{B}(c_i, 3R)$ for all complete $\mathcal{B}(c_i, 3R)$.

Compute $d^{\text{est}}_{T_i}(x, c_i) = \mathsf{Median}\{\widetilde{d}(x, y) \mid y \in T_i\}$ for all T_i.

Step 4. Call x covered by $\mathcal{B}(c_i, 3R)$ if $d^{\text{est}}_{T_i}(x, c_i) \leq 6R$ and assign x to c_i.

end

Remove all covered points form $\mathcal{X}$.

end

We will prove the approximation guarantee for the set of candidate centers generated by algorithm 4 in the following order, corresponding to respective steps in algorithm 4.

Step 1. Centers returned by algorithm 3 on S can be used to identify heavy balls in $\mathcal{B}^*_{\mathcal{H}}$.

Step 2. T provides sufficient points in heavy balls identified in Step 1 for distance estimation (heavy balls are complete).

Step 3. Distance estimated using complete balls is accurate enough for $O(1)$ approximation.

Step 4. Using heavy balls and distances from step 3, we cover a constant fraction of points each iteration.

Step 1. The combined objective of step 1 and 2 in the algorithm is to identify the heavy balls and the points belonging to these heavy balls. We by start sampling two sets of points (S and T), each of size $O(k \log n)$ uniformly at random from $\mathcal{X}$ and querying strong oracle on them. We first make following observations that help formalize the process of identification of heavy balls.

Observation B.3 *Let n be the number of points in $\mathcal{X}$ and let S be a set of $100 \cdot k \log n$ points sampled uniformly at random from $\mathcal{X}$. Then, with high probability, for every $\mathcal{B}^*(c^*_i, \mathit{OPT}) \in \mathcal{B}^*_H$, there are at least $\log n$ points in S covered by $\mathcal{B}^*(c^*_i, \mathit{OPT})$, i.e. $|\mathcal{B}^*(c^*_i, \mathit{OPT}) \cap S| \geq \log n$.*

Proof As each heavy $\mathcal{B}^*_i(c^*_i, \mathsf{OPT}) \in \mathcal{B}^*_H$ covers at least $n/10k$ points in $\mathcal{X}$, in expectation we have at least $10 \log n$ points in S that are covered by $\mathcal{B}^*_i(c^*_i, \mathsf{OPT})$. By Chernoff bounds, for $\mathcal{B}_i^*(c_i^*, \mathsf{OPT})$

$$\Pr\left(|\mathcal{B}_i^*(c_i^*, \mathsf{OPT}) \cap S| \leq \log n\right) \leq \exp\left(-\frac{81}{200} \cdot 10 \log n\right) \leq n^{-4}.$$

Using union bound over at most k heavy balls gives us that each $\mathcal{B}^*(c_i^*, \mathsf{OPT}) \in \mathcal{B}_H^*$ covers at least $\log n$ points in S w.p. $1 - k/n^4$. ■

This tells us that S has at least $\log n$ points from each heavy ball in optimal solution w.h.p. We now use algorithm 3 to cover all points in S with at most k centers using radius R, and use the centers to form heavy balls. Let C be the set of centers that are returned by algorithm 3 for set S. For each $c_i \in C$, consider the balls $\mathcal{B}(c_i, 3R)$. As $2R + \mathsf{OPT} \leq 3R$, using $3R$ as radius for $\mathcal{B}(c_i, 3R)$ ensures that the union of these balls cover all points that are covered by $\mathcal{B}_H^*$ ($\frac{9n}{10}$ points in $\mathcal{X}$). We now define $\mathcal{B}_{\mathcal{H}}$ as the set of $\mathcal{B}(c_i, 3R)$ that are heavy by definition 14 among all $c_i \in C$, i.e.,

$$\mathcal{B}_H = \{\mathcal{B}(c_i, 3R) \mid c_i \in C,\, |\mathcal{B}(c_i, 3R)| \geq \frac{n}{10k}\}. \tag{1}$$

Following similar argument as Claim B.1, non-heavy balls $\mathcal{B}(c_i, 3R)$ covers at most $n/10$ points in $\mathcal{B}_H^*$, which equates to $\mathcal{B}_{\mathcal{H}}$ covering at least $(\frac{9n}{10} - \frac{n}{10}) = \frac{8n}{10}$ points in $\mathcal{X}$.

Step 2. We now argue that T has enough points for each heavy ball in $\mathcal{B}_H$, which can be used for distance estimation for the remainder of points in $\mathcal{X}$. We show that it suffices to have $O(\log n)$ points in T that are covered by a heavy ball to get a good estimate to its center. Following this, we call $\mathcal{B}(c_i, 3R)$ '*complete*' if T contains at least $100 \log n$ points that are covered by $\mathcal{B}(c_i, 3R)$. Note that this can be verified by the algorithm as we use strong oracle queries for S and T. If any of the points from S or T are uncovered, we just add them to the list of candidate centers (complete balls centers), and since we run greedy ball carving at the end on candidate centers they will be covered. We start by proving the following claim.

Claim B.4 *For the set of centers C returned by algorithm 3 on S and $|T| = 2000 \cdot k \log n$ set of points sampled uniformly at random from $\mathcal{X}$, every $\mathcal{B}(c_i, 3R) \in \mathcal{B}_H$ for $c_i \in C$ is complete w.h.p.*

Proof The argument is similar to Claim B.3, where we first show for a particular $c_i \in C$ and then union bound over at most k such centers. From eq (1), in expectation T has at least $200 \log n$ points covered by a heavy $\mathcal{B}(c_i, 3R)$. Using Chernoff bounds,

$$\Pr\left(|\mathcal{B}_i(c_i, 3R) \cap T| \leq 100 \log n\right) \leq \exp\left(-\frac{200}{8} \log n\right) \leq n^{-25}.$$

Using union bound over at most k centers, w.p. $1 - (k/n^{25})$ every heavy $\mathcal{B}(c_i, 3R)$ is complete. ■

Step 3. At this stage we have at most k complete balls $\mathcal{B}(c_i, 3R)$, each with $O(\log n)$ points. We now use median of weak oracle queries to these points in each heavy ball for distance estimation of remaining points $x \in \mathcal{X} \setminus S \cup T$. For each $x \in \mathcal{X} \setminus \{S \cup T\}$ and each complete $\mathcal{B}(c_i, 3R)$, let $d^{\text{est}}_{T_i}(x, c_i) = \mathsf{Median}\{\widetilde{d}(x, q) \mid q \in T_i\}$ for $T_i = T \cap \mathcal{B}(c_i, 3R)$ and $\widetilde{d}(v, q)$ denotes the weak oracle queries. These distance estimates form the 'workhorse' of sample and cover for assigning points to heavy balls. The following lemma quantifies the accuracy of the estimated distance.

Lemma 16 *Fix any point $u \in \mathcal{X}$, radius $r \geq 0$, and set $U \subset \mathcal{B}(u, r)$ with $|U| = \Omega(\log n)$. For any $x \in \mathcal{X}$, define $d^{est}_U(x, u) = \mathsf{Median}\{\widetilde{d}(x, q) | q \in U\}$. Then with high probability, for all $x \in \mathcal{X}$ we have $|d^{est}_U(x, u) - d(x, u)| \leq r$.*

Proof Recall that the weak oracle returns arbitrary value for $\widetilde{d}(x, y)$ w.p. $1/3$, independently for each $x, y \in \mathcal{X}$. For $|U| \geq c \cdot \log n$ for a sufficiently large constant c, in expectation $\frac{2}{3}c \cdot \log n$ weak oracle calls are not corrupted. For using median as an estimate, we only need more than half queries to not be corrupted. Let $\mathbf{X}_u$ denote the number of uncorrupted weak oracle queries U. Using Chernoff bounds,

$$\Pr\left(\mathbf{X}_u \leq \frac{c}{2}\log n\right) \leq \exp\left(-\frac{c\log n}{16 \cdot 3}\right) \leq n^{-c'}$$

Thus, for sufficiently large c, with high probability the median distance lies between $\min_{u \in U} d(x, u)$ and $\max_{u \in U} d(x, u)$. It follows from union bound over at most n points and triangle inequality that $|d^{\text{est}}_U(x, u) - d(x, u)| \leq r$ for all $x \in \mathcal{X}$.

∎

Considering Lemma 16 for estimating distance to centers of complete balls, for each complete $\mathcal{B}(c_i, 3R)$ we have $200 \cdot \log n$ points in T. For a particular complete $\mathcal{B}(c_i, 3R)$ we have w.p. n^{-4}, $|d^{\text{est}}_{T_i}(x, c_i) - d(x, c_i)| \leq 3R$ for $x \in \mathcal{X} \setminus \{S \cup T\}$. Taking union bound over at most k complete balls and distance estimates for at most n points, the estimation guarantee holds w.p. $1 - \frac{k}{n^3}$ for all distance estimates of $x \in \mathcal{X} \setminus \{S \cup T\}$ to all complete balls $\mathcal{B}(c_i, 3R)$.

Step 4. With the distance estimates from Lemma 16, the algorithm assigns x to a complete $\mathcal{B}(c_i, 3R)$ if $d^{\text{est}}_{T_i}(x, c_i) \leq 6R$, with ties broken arbitrarily. Recall that union of all complete $\mathcal{B}(c_i, 3R)$ balls cover $8/10$ fraction of points in $\mathcal{X}$. As the distance estimates are accurate upto $\pm 3R$ and we assign if distance to centers is at most $6R$, at each step $8/10$ fraction of points are covered by the complete balls. This is achieved by making $O(k \log n)$ strong oracle queries (recall the algorithm uses strong oracle queries only for set S and T).

Thus, at each iteration we cover a constant fraction of points in $\mathcal{X}$ using at most k centers. For covering all points in $\mathcal{X}$, the recursion will proceed for $O(\log n)$ rounds, generating $O(k \log n)$ centers. We now prove the approximation guarantee obtained by these set of centers.

Lemma 17 *Suppose $2\mathsf{OPT} \leq R \leq 2 \cdot (1 + \varepsilon)\mathsf{OPT}$, then with high probability, Algorithm 4 computes a set of $O(k \log n)$ centers and the clustering assignment for each point $x \in \mathcal{X}$ using $O(k \log^2 n)$ strong oracle queries, $O(nk \log^2 n)$ weak oracle queries such that each point is at most $12 \cdot (1 + \varepsilon)\mathsf{OPT}$ from its respective center.*

Proof At each iteration, algorithm 4 produces at most k centers that cover constant fraction of points in $\mathcal{X}$ and the procedure runs for $O(\log n)$ iterations. The total number of strong oracle queries we make is $(\log n \cdot (|S| + |T|)) = O(k \log^2 n)$. For number of weak oracle queries, note at each iteration we have $O(\log n)$ points in at most k complete balls. For estimating distances of at most n remaining points at each iteration we make $O(nk \log n)$ weak oracles queries. For $O(\log n)$ such iterations, we make total $O(nk \log^2 n)$ weak oracle queries.

At the end of recursion, any point $x \in \mathcal{X}$ is at most $6R$ from respective center in set of $O(k \log n)$ centers, and $2\mathsf{OPT} \leq R \leq 2 \cdot (1+\varepsilon)\mathsf{OPT}$, this gives us $12 \cdot (1+\varepsilon)$-approximate solution.

■

In order to obtain a solution for k-center problem, we now run algorithm 3 on the set of $O(k \log n)$ centers (also uncovered points with strong oracle query, if any, from S and T at any iteration).

From Claim B.2 we obtain at most k centers. Using the approximation guarantee from Lemma 17 and additional iterations for estimation of R, we now wrap up the proof for theorem 1.

Proof of Theorem 1 We first analyze the total number of strong and weak oracle queries used by the algorithm, by looking at additional overhead from estimating R. First, while there are $O(\frac{1}{\varepsilon} \log n)$ possible values of R to try, instead of trying each in increasing order as in theorem 1, we claim that we can binary search to find the correct value. Specifically, for the range $r \in \{1, (1+\varepsilon), (1+\varepsilon)^2, \ldots, \Delta\}$, we simply need to find a value of R such that running algorithm 2 on that value of $R/(1+\varepsilon)$ returns more than k centers, (thus implying that $R < 2(1+\varepsilon)\mathsf{OPT}$), and such that running algorithm 2 on R returns at most k centers (thus implying that we get a valid solution from algorithm 2 with a value of R). Importantly, this need not be the smallest R such that the above occurs. Thus, we can find such an R via binary search, which requires a total of $O(\log(\frac{\log n}{\varepsilon}))$ rounds of running algorithm 2. Since each run of algorithm 2 uses $O(k \log^2 n)$ strong oracle queries and $O(nk \log^2 n)$ weak oracle queries, after the binary search it follows that the total query complexity is $O(k \log^2 n \log\left(\frac{\log n}{\varepsilon}\right))$ strong oracle queries and $O(nk \log^2 n \log\left(\frac{\log n}{\varepsilon}\right))$ weak oracle queries.

Now we look at the approximation factor. Let $R = (1+\varepsilon)^j$ be such a value found via the above binary search. As above, we have that $R \leq 2(1+\varepsilon)\mathsf{OPT}$, and that we obtained a valid solution from algorithm 2 with a value of R. For covering the points, since we assign points to a complete $\mathcal{B}(c_i, 3R)$ with distance to center at most $6R$ and later run algorithm 3 with threshold R on $O(k \log n)$ centers, any point is at most $7R$ away from the center which gives a $14(1+\varepsilon)$-approximate solution (since $R \leq 2(1+\varepsilon)\mathsf{OPT}$).

For analyzing the run-time of our algorithm, we first look at one iteration of our algorithm. Note, greedy ball carving on a set of $O(k \log n)$ points for at most k centers takes $O(k^2 \cdot \operatorname{polylog} n)$ time. Then, conditioning on the high probability event of Claim B.4, for at most k complete balls with $O(\log n)$ points each, estimating median of remaining points to respective centers takes $O(nk \cdot \operatorname{polylog} n)$ time. At most $O(\log n)$ such iterations coupled with iterations for estimating R, the total run time is $O\left(k^2 \cdot \operatorname{polylog} n \log\left(\frac{\log n}{\varepsilon}\right) + nk \cdot \operatorname{polylog} n \log\left(\frac{\log n}{\varepsilon}\right)\right) = \tilde{O}\left(nk \log\left(\frac{1}{\varepsilon}\right)\right)$.

Theorem 1 ■

Appendix C. Analysis of the k-means and k-median algorithms in section 3

In this section, we present the proof of our main algorithmic result for k-means clustering (Theorem 2). We now proceed to analyze Algorithm 1. Algorithm 1 requires an approximation $\widetilde{\mathsf{OPT}}$,of the optimal cost OPT. We will first assume we have such a value $\widetilde{\mathsf{OPT}}$ satisfying $2^i \leq \widetilde{OPT} \leq 2^{i+1}$, and later describe how we can find it via binary search. Specifically, we will terminate any run of Algorithm 1 whenever it samples more than $1800k\log^2 n$ points to add to S, and conduct a binary search by maintaining upper and lower bounds of the indices i. In the end, we output the subroutine induced by an $i^\star$ such that (i). the subroutine with $2^{i^\star}$ returns a clustering (i.e. did not terminate) and (ii). the subroutine with $2^{i^\star-1}$ is terminated without return. We then set $\widetilde{\mathsf{OPT}} = 2^{i^\star}$. We will show that this value of $\widetilde{\mathsf{OPT}}$ satisfies the desired properties. As such, our analysis focus on the case when $\mathsf{OPT} \leq \widetilde{\mathsf{OPT}} \leq 2\mathsf{OPT}$.

For the simplicity of presentation, we focus on the analysis of k-median as it does not involved the square terms on the distances. We show in Remark 21 how the analysis also works for the k-means clustering with a slightly larger constant factor. For convenience, we first restate the definition of the heavy ball distance $Q(q, S)$, as it will play a central role in our analysis. We also give an illustration of the computation of the heavy-ball distance in Figure 1.

Definition 8 (Heavy-ball distance) *Fix the metric space $(\mathcal{X}, d)$, let $q \in \mathcal{X}$ be a point and $S \subseteq \mathcal{X} \setminus \{q\}$ be a set of points such that $|S| \geq 100\log n$. Then for any $x \in S$ and radius $r > 0$, we set $Q(q, S;\ x, r) = \textsf{Median}\{\widetilde{d}(q, z) \mid z \in S \cap \mathcal{B}(x, r)\} + 6 \cdot r$, and define the heavy-ball distance as*

$$Q(q, S) = \min_{\substack{x \in S, r>0 \\ |\mathcal{B}(x,r)\cap S| \geq 100\log(n)}} Q(q, S;\ x, r).$$

To proceed with the analysis, we also introduce some self-contained notations used in this analysis. We let $c_1^*, \cdots, c_k^*$ be the centers for the optimal k-median solution, and we let $r_1^*, r_2^*, \cdots, r_k^*$ be the *average* cost of the points in each cluster, i.e.,

$$r_i^* = \frac{1}{|\{x \mid \mathcal{C}(x) = c_i^*\}|} \cdot \sum_{x\, s.t.\, \mathcal{C}(x) = c_i^*} d(x, c_i^*).$$

Based on this, we can define $\mathcal{B}_i^j$ as the ball centered at c_i^* and with distance at most $2^j r_i^*$. We further define A_i^j to be the set of points with distance $(2^{j-1} r_i^*,\ 2^j r_i^*]$ from the center of C_i, i.e. an "annulus" between $\mathcal{B}_i^{j-1}$ and $\mathcal{B}_i^j$. We also define the following event:

$\mathcal{E}_{A_{i,j}\text{-heavy}}$: the set A_i^j has at least $100\log n$ points in S, i.e. $|A_i^j \cap S| \geq 100\log n$.

We let the points being sampled in S before $\mathcal{E}_{A_{i,j}\text{-heavy}}$ (also denote as $\neg\mathcal{E}_{A_{i,j}\text{-heavy}}$) as $S_{i,j}^{\text{light}}$, and let its complement be $S_{i,j}^{\text{heavy}}$, i.e. the set of points sampled after $\mathcal{E}_{A_{i,j}\text{-heavy}}$. In what follows, we will first show the approximation guarantees and the number of centers in S.

Lemma 18 *Suppose $\mathsf{OPT}_{k\text{-median}} \leq \widetilde{\mathsf{OPT}} \leq 2\mathsf{OPT}_{k\text{-median}}$. Then, with probability at least $1 - \frac{1}{n^3}$, the clustering outputted by S in Algorithm 1 gives an $O(1)$-approximation of the $\mathsf{OPT}_{k\text{-median}}$.*

Proof For each of the set A_i^j, we analyze the cost it pays in the formation of S by looking at $\neg\mathcal{E}_{A_{i,j}\text{-heavy}}$ (before A_i^j becomes heavy) and $\mathcal{E}_{A_{i,j}\text{-heavy}}$ (after A_i^j becomes heavy), respectively as in Claim C.1 and Claim C.2.

Claim C.1 *For each set A_i^j, with probability at least $1 - \frac{1}{n^4}$, the cost induced by all points in $A_i^j \cap S_{i,j}^{light}$ is at most $200f \cdot \log n$.*

Proof For any point $x \in A_i^j \cap S_{i,j}^{\text{light}}$, we define random variable X_x as the indicator random variable for x to be sampled. Since we sample each point with probability at most $Q(x, S)/f$, there is

$$\mathbb{E}\left[X_x\right] \leq Q(x, S_{:x})/f,$$

where we use $S_{:x}$ to denote the sampled set before x is visited. Suppose $\mathcal{E}_{A_{i,j}\text{-heavy}}$ happens after sampling N points from A_i^j, which means up till the $N-1$ point, we have

$$\mathbb{E}\left[\sum_{i\in[N-1]} X_{x_i}\right] \leq \sum_{i\in[N-1]} Q(x, S_{:x_i})/f < 100 \log n.$$

The last inequality holds since we condition on the event that the number of sampled points is (deterministically) less than $100 \log n$. Similarly, we can also get a lower bound on the expectation by using the fact that we are only one point short of reaching $100 \log n$ (assuming $n \geq 10$).

$$\mathbb{E}\left[\sum_{i\in[N-1]} X_{x_i}\right] \geq 99 \log n.$$

On the other hand, note that if a point X_x is not sampled, the cost induced by X_x is exactly $Q(x, S_{:x_i})$. As such, for the induced cost of the points in $A_i^j \cap S_{i,j}^{\text{light}}$ to be more than $200f \cdot \log n$, there must be

$$\sum_{i\in[N-1]} Q(x, S_{:x_i}) > 200f \cdot \log n.$$

Comparing the two inequalities, and using the fact that $\sum_{i\in[N-1]} X_{x_i}$ is a summation of $0/1$ random variables, we have that for the second inequality to happen, the probability is

$$\begin{aligned}
\Pr\left(\text{cost induced by } A_i^j \cap S_{i,j}^{\text{light}} > 200f \cdot \log n\right) &= \Pr\left(\sum_{i\in[N-1]} Q(x, S_{:x_i}) > 200f \cdot \log n\right) \\
&\leq \Pr\left(\sum_{i\in[N-1]} X_{x_i} \geq 2 \cdot \mathbb{E}\left[\sum_{i\in[N-1]} X_{x_i}\right]\right) \\
&\leq \exp\left(-\frac{1}{3} \cdot \mathbb{E}\left[\sum_{i\in[N-1]} X_{x_i}\right]\right) \\
&\qquad\qquad \text{(by multiplicative Chernoff)} \\
&\leq \frac{1}{n^7},
\end{aligned}$$

as desired.

Claim C.1 ■

We now turn to the cost of the points in A_i^j after the set becomes heavy, i.e. conditioning on $\mathcal{E}_{A_{i,j}\text{-heavy}}$ happens.

Claim C.2 *For each set A_i^j, with probability at least $1 - \frac{1}{n^4}$, the cost induced by all points in $A_i^j \cap S_{i,j}^{heavy}$ is at most 13-multiplicative of the cost induced by $A_i^j \cap S_{i,j}^{heavy}$ on the optimal clustering.*

Proof Note that the moment $\mathcal{E}_{A_{i,j}\text{-heavy}}$ happens, the ball $\mathcal{B}_i^j$ becomes heavy as well. As such, by the same argument we used in Lemma 16, the cost of adding any $x \in A_i^j \cap S_{i,j}^{\text{heavy}}$ to its nearest heavy-ball cluster is at most $Q(x, S) \leq d(x, c_i^*) + 6 \cdot 2^j r_i^*$ with probability at least $1 - \frac{1}{n^4}$. Furthermore, we have $d(x, c_i^*) \geq 2^{j-1} r_i^*$ by the fact that $x \in A_i^j$. Therefore, we can charge the cost of x induced in S to at most $13 \cdot d(x, c_i^*)$. This implies a 13-multiplicative approximation for all points in $A_i^j \cap S_{i,j}^{\text{heavy}}$, as desired.

Claim C.2 ■

We now finalize the proof of Lemma 18. We first argue that we can apply Claim C.1 to *all* A_i^j sets. Note that there are at most $O(k \log n) \leq O(n \log n)$ such sets, as for any cluster c_i^*, there is no point whose distance is more than $n\, r_i^*$. Therefore, we can apply a union bound to conclude that with probability at least $1 - \frac{1}{n^5}$, the bound of Claim C.1 applies to all such A_i^j sets. Collectively, they induce at most $200 \cdot fk \log^2 n \leq 10\, \widetilde{OPT} \leq 20\, \mathsf{OPT}_{\text{k-median}}$. Furthermore, by Claim C.2, the total cost we induce on $A_i^j \cap S_{i,j}^{\text{heavy}}$ for all i, j is at most $13\mathsf{OPT}_{\text{k-median}}$. As such, the total cost induced by S is at most $O(1) \cdot \mathsf{OPT}_{\text{k-median}}$, as desired. ■

We now turn to the analysis of the number of centers that are ever sampled in S.

Lemma 19 *Suppose $OPT_{k\text{-median}} \leq \widetilde{OPT} \leq 2OPT_{k\text{-median}}$. Then, with probability at least $1 - \frac{1}{n^3}$, the set S output in Algorithm 1 has at most $O(k \log^2 n)$ points.*

Proof We again analyze the number of sampled points before and after the event $\mathcal{E}_{A_{i,j}\text{-heavy}}$. Note that before $\mathcal{E}_{A_{i,j}\text{-heavy}}$, we deterministically only provide at most $100\log n$ centers from A^j_i. After $\mathcal{E}_{A_{i,j}\text{-heavy}}$, the ball $\mathcal{B}^j_i$ becomes heavy. As such, define X_x be the indicator random variable for sampling x from A^j_i, we can condition on the high probability event of Claim C.2, and argue that with probability at least $1-\frac{1}{n^3}$, there is

$$\begin{aligned}\mathbb{E}\left[X_x \mid \mathcal{E}_{A_{i,j}\text{-heavy}}\right] &\leq 260\cdot \frac{d(x,c^*_i)\,k\,\log^2 n}{\widetilde{\mathsf{OPT}}}\\ &\leq 260\cdot \frac{d(x,c^*_i)\,k\,\log^2 n}{\mathsf{OPT}_{\text{k-median}}}.\end{aligned}$$

Therefore, we can add up all the centers and their j values, which gives us

$$\begin{aligned}\mathbb{E}\left[\sum_{i,j}\sum_{x\in A^i_j} X_x \mid \mathcal{E}_{A_{i,j}\text{-heavy}}\right] &= \sum_{i,j}\sum_{x\in A^i_j}\mathbb{E}\left[X_x \mid \mathcal{E}_{A_{i,j}\text{-heavy}}\right] && \text{(linearity of expectation)}\\ &\leq 260\cdot\frac{k\log^2 n}{\mathsf{OPT}_{\text{k-median}}}\cdot\sum_{i,j}\sum_{x\in A^i_j} d(x,c^*_i)\\ &= 260\cdot k\log^2 n. && \left(\textstyle\sum_{i,j}\sum_{x\in A^i_j} d(x,c^*_i)=\mathsf{OPT}_{\text{k-median}}\right)\end{aligned}$$

As such, we can bound the *expectation* of the total number of points that are ever sampled as:

$$\begin{aligned}\mathbb{E}\left[\sum_{i,j}\sum_{x\in A^i_j} X_x\right] &\leq \mathbb{E}\left[\sum_{i,j}\sum_{x\in A^i_j} X_x \mid \mathcal{E}_{A_{i,j}\text{-heavy}}\right]+\mathbb{E}\left[\sum_{i,j}\sum_{x\in A^i_j} X_x \mid \neg\,\mathcal{E}_{A_{i,j}\text{-heavy}}\right]\\ &\leq 260\cdot k\log^2 n+\sum_{i,j}100\log n\\ &\leq 360\cdot k\log^2 n. && \text{(there are only } k\log n \text{ possible } A^j_i \text{ sets)}\end{aligned}$$

Finally, we analyze the concentration of the number of sampled points in S. Note that using Claim A.1, we can verify that the X_x indicator random variables are negatively correlated, i.e. $\Pr(X_x=1\mid X_u=1)\leq \Pr(X_x=1)$ for $u\neq x$. If $\sum_{i,j}\sum_{x\in A^i_j} X_x<100k\log^2 n$, we deterministically obtain the desired bound; on the other hand, assuming $\sum_{i,j}\sum_{x\in A^i_j} X_x\geq 100k\log^2 n$, we have

$$\begin{aligned}\Pr\left(\sum_{i,j}\sum_{x\in A^i_j} X_x\geq 5\cdot\mathbb{E}\left[\sum_{i,j}\sum_{x\in A^i_j} X_x\right]\right) &\leq \exp\left(-\frac{16}{6}\cdot\mathbb{E}\left[\sum_{i,j}\sum_{x\in A^i_j} X_x\right]\right)\\ &\qquad\text{(Chernoff bound for negatively correlated random variables, Proposition 12)}\\ &\leq \frac{1}{n^3},\end{aligned}$$

as desired. ■

By Lemma 18 and Lemma 19, after the procedure that produces S terminates, we have set set of at most $O(k \log^2 n)$ centers and an $O(1)$-approximation of the optimal k-median objective. We now describe how this result leads to the k-median solution with exactly k centers. To this end, we introduce the following standard result for approximating k-median on coresets (see e.g. Guha et al. (2000)) and a proof for completeness.

Proposition 20 (Guha et al. (2000)) *Let $\{c_1, \cdots, c_m\}$ be a set of centers that achieves α-approximation of the k-median (resp. k-means) objective, and let $w_1, \cdots, w_m$ be the number of points contained in each cluster induced by $\{c_i\}_{i=1}^m$. Furthermore, let $\{\tilde{c}_1, \tilde{c}_2, \cdots, \tilde{c}_k\}$ be a β-approximate k-median (resp. k-means) on the weighted points $\{c_1, \cdots, c_m\}$. Then, the clustering induced by $\{\tilde{c}_1, \tilde{c}_2, \cdots, \tilde{c}_k\}$ gives an $O(\alpha + \beta)$-approximation of $\mathsf{OPT}_{\text{k-median}}$ (resp. $\mathsf{OPT}_{\text{k-means}}$).*

Proof For any point $x \in \mathcal{X}$, let $c(x)$ be its clustering center in $\{c_i\}_{i=1}^m$ and $\tilde{c}(x)$ be its clustering center in $\{\tilde{c}_j\}_{j=1}^k$. If $c(x) = \tilde{c}(x)$, the cost induced by x remains $d(x, c(x))$. On the other hand, if $c(x) \neq \tilde{c}(x)$, the cost induced by x is at most $d(c(x), \tilde{c}(x)) + d(x, c(x))$. Furthermore, note that if we let OPT_{ws} be optimal cost of the weighted cost of clustering on $\{\tilde{c}_j\}_{j=1}^k$, and let $\tilde{c}(\cdot)$ and $c^*(\cdot)$ be the functions that maps 1). points in $\{c_i\}_{i=1}^m$ to the center in the weighted clustering and 2). points in $\mathcal{X}$ to the optimal k-median clustering, respectively. As such, we have

$$\begin{aligned}
\mathsf{OPT}_{ws} &= \sum_{i=1}^{m} w_i \cdot d(c_i, \tilde{c}(c_i)) \\
&= \sum_{i=1}^{n} d(c_i, \tilde{c}(c_i)) \\
&\qquad \text{(duplicating each } c_i \text{ for } w_i \text{ times and map all of them to the corresponding center)} \\
&\leq \sum_{i=1}^{n} d(c_i, c^*(c_i)) + d(\tilde{c}(c_i), c^*(\tilde{c}(c_i))) \\
&\leq 2\mathsf{OPT}_{\text{k-median}}.
\end{aligned}$$

Therefore, we can bound the total cost with

$$\begin{aligned}
\sum_{x \in \mathcal{X}} d(x, \tilde{c}(x)) &\leq \sum_{x \in \mathcal{X}} d(x, c(x)) + \sum_{x \in \mathcal{X}} d(c(x), \tilde{c}(x)) \\
&\leq \alpha \cdot \mathsf{OPT}_{\text{k-median}} + \beta \cdot \mathsf{OPT}_{ws} \\
&\leq (\alpha + 2\beta) \cdot \mathsf{OPT}_{\text{k-median}},
\end{aligned}$$

as desired.

Finally, for the k-means case, we need to replace $d(\cdot, \cdot)$ with $d^2(\cdot, \cdot)$; although this is no longer a metric, we can use the approximate triangle inequality that

$$d^2(x, z) \leq 2(d^2(x, y) + d^2(y, z)).$$

As such, by replacing the k-median clustering centers with the k-means ones and use $d^2(\cdot,\cdot)$, we can get

$$\begin{aligned}\sum_{x\in\mathcal{X}} d^2(x,\tilde{c}(x)) &\le 2\cdot\sum_{x\in\mathcal{X}} d^2(x,c(x)) + 2\cdot\sum_{x\in\mathcal{X}} d^2(c(x),\tilde{c}(x))\\ &\le (2\alpha+8\beta)\cdot \mathsf{OPT}_{\text{k-means}},\end{aligned}$$

as desired. ■

Proof of Theorem 2 Conditioning on the right guess of $\mathsf{OPT}_{\text{k-median}} \le \widetilde{\mathsf{OPT}} \le 2\mathsf{OPT}_{\text{k-median}}$, by Lemma 18 and Lemma 19, in the construction of set S we sample at most $O(k\log^2 n)$ points and produce an $O(1)$-approximation. This also implies we make at most $O(k\log^2 n)$ strong oracle SO queries. To bound the number of weak oracle queries, note that in each iteration of x_i, we only need to query the distances between the x_i and the points in S, which is at most $O(nk\,\log^2 n)$, and our post-processing of the points in S does not involve any additional oracle queries.

To see the time efficiency, observe that we can maintain a heap for each point in S to represent the balls of size $O(\log n)$. As such, when x_i is sampled and a strong oracle query SO is added, it takes $O(|S|\log n)$ time to insert the value and remove the largest one from the heap. Therefore, conditioning on the high probability event of Lemma 19, the updates of the heavy balls takes at most $O(\sum_{|S|=1}^{k\log^2 n}|S|\log n) = O(k^2\operatorname{polylog} n)$ time. On the other hand, if x_i is not sampled, we need to estimate the median from x_i to every ball in S, and it take $O(|S|\log n)$ time by the heap structure. Therefore, the estimation of heavy ball nearest distance takes $O(n\,|S|\log n) \le O(nk\operatorname{polylog} n)$ time across the process. Finally, we can run $O(1)$-approximate k-median algorithms in $O(|S|k)$ time by Mettu and Plaxton (2004), which gives us $O(k^2\text{polylog}\, n)$ time by the size bound of $|S|$. In total, this gives us the desired $O(nk\operatorname{polylog} n)$ time.

For the approximation guarantee, by Lemma 18 and Proposition 20, we can run an $O(1)$-approximate k-median (resp. k-means) algorithm on the $O(1)$-approximate coreset of S [5]. The correctness is guaranteed since we know the *exact* distance between the points in S (by the SO queries). Therefore, we get an $O(1)$ approximation of the optimal clustering cost.

Finally, when running with unknown $\widetilde{OPT}$, we break a run whenever it samples more than $1800k\log^2 n$ points in the construction of S, and output the run with the $\widetilde{OPT}$ value that (i). is not break and (ii). is next to a run with $\widetilde{OPT}/2$ that breaks. Thus, we can binary search for such a value of $\widetilde{OPT}$ (as discussed in Section A.2). Using that $\Delta \le \operatorname{poly}(n)$, this gives us $O(\log\log n)$ overhead for the queries and running time. As such, it results in at most $O(k\log^2 n\log\log n)$ strong oracle SO queries, at most $O(nk\log^2 n\log\log n)$ weak oracle WO queries, and $O(nk\operatorname{polylog} n)$ time.

Theorem 2 ■

Remark 21 *We now describe the changes needed to generalize algorithm 1 from the k-medians objective to k-means. Specifically, since the cost measure in k-means is* $\sum_{x\in\mathcal{X}} d^2(x,\mathcal{C}(x))$

5. If we do not care about the time efficiency, we can run the state-of-the-art polynomial-time algorithm by Ahmadian et al. (2017); we can even run a brute-force algorithm to search for the minimum-cost clustering

instead of $\sum_{x\in\mathcal{X}} d(x,\mathcal{C}(x))$*, it will suffice to change our "distance" measure to* d^2*. This can be dome by making the modification to the sampling probability of* x_i*: instead of using* $\min\{1, Q(x_i, S)/f\}$*, we will use the sampling probability of* $\min\{1, Q^2(x_i, S)/f\}$*. The only downstream change that occurs is that we no longer can apply triangle inequality, since* $d^2(\cdot,\cdot)$ *is no longer a metric. However, we can always employ the approximate triangle inequality of* $d^2(x,z) \le 2\cdot(d^2(x,y)+d^2(y,z))$ *(see Proposition 20 for the usage). Note that the triangle inequality is only used in Claim C.2 and Proposition 20, and the rest of the algorithm and the analysis proceeds exactly as in the* k*-median case (Lemma 19 is affected by the high probability event in Claim C.2, but the analysis itself does not use triangle inequality). Substituting approximate triangle inequality for the triangle inequality induces an additional constant factor into the objective, which does not effect our overall* $O(1)$*-approximation.*

Appendix D. Generalizing Our Clustering Algorithms to Larger δ

We used the fixed corruption probability $\delta = \frac{1}{3}$ in our proofs of our clustering algorithms for clarity of presentation. However, we remark that our clustering algorithm works for arbitrary $\delta < \frac{1}{2}$ by scaling up the number of weak and strong oracle queries by a factor of $(\frac{1}{1/2-\delta})^2$. To see this, note that we only used the corruption probability in the median estimation, and our goal is show that in a fixed set S of points, with high probability, for any $x \in \mathcal{X}$, at least half of the distances between x and $y \in S$ are not corrupted. In the Chernoff bound calculation, if the corruption probability is δ, there are in expectation $\mathbb{E}[X] = (1-\delta)|S|$ distances between $x \in \mathcal{X}$ and $y \in S$ preserved for a fixed S. As such, the calculation of Chernoff bound becomes

$$\begin{aligned}
\Pr\left(X < \frac{|S|}{2}\right) &= \Pr\left(X < \frac{1/2}{1-\delta}\cdot\mathbb{E}[X]\right)\\
&= \Pr\left(X < (1-\frac{1/2-\delta}{1-\delta})\cdot\mathbb{E}[X]\right)\\
&\le \Pr\left(X < (1-(1/2-\delta))\cdot\mathbb{E}[X]\right)\\
&\le \exp\left(-\frac{(1/2-\delta)^2\cdot\mathbb{E}[X]}{3}\right)\\
&\le \exp\left(-\frac{(1/2-\delta)^2\cdot|S|}{6}\right),
\end{aligned}$$

where the last two inequalities used the condition of $\delta < \frac{1}{2}$. As such, the condition of $|S| = O(\frac{1}{(1/2-\delta)^2}\cdot\log n)$ suffices to keep the statement true with high probability.

Appendix E. Minimum Spanning Tree in the Weak-Strong Oracle Model

We now present the full proofs from Section 4. We begin with the proof of Proposition 22.

First, for any integer k, we define H_k to be the unique complete binary labeled tree on $k+1$ vertices such that the level-order traversal of H_k is $0,1,2,\ldots,k$. In other words, H_k is a complete binary tree where the zero-th level contains just the root, labeled 0, the first level contains vertices labeled $1,2$ (left to right ordered), and second the labels $3,4,5,6$, and so on.

Define the mapping $\varphi : \mathbb{Z}_{\geq 1} \to \mathbb{Z}_{\geq 0}$ by $\phi(i) = j$ where j is the label of the parent of i in the graph H_k (for any $k \geq i$): namely, $\varphi(i) = \lceil i/2 \rceil - 1$.

Algorithm 5 **Bounded-Degree Tree Transformation**

Input: Rooted tree T over a metric space $(\mathcal{X}, d)$.
Output: Rooted tree $\hat{T}$ with degree at most 5 with $w_d(\hat{T}) \leq 2w_d(T)$.
Init: $\hat{T} = (V, \hat{E})$, where $\hat{E} = \{\}$ is an empty edge set.

for *every* $u \in T$ **do**
  Let C_u be the set of children of u in T, and set $k = |C_u|$.
  if $k \leq 2$ **then**
    Add directed edge (u, v) to $\hat{E}$ for every $v \in C_u$
  end
  else
    Order the children $C_u = \{x_1, \ldots, x_k\}$ so that $d(u, x_1) \leq d(u, x_2) \leq \cdots \leq d(u, x_k)$, and define $x_0 = u$.
    For each $i = 1, 2, \ldots, k$, add the directed edge $(x_{\varphi(i)}, x_i)$ to $\hat{E}$.
  end
  Remove all covered points form V.
end

Proposition 22 *Fix any n-point metric space $(\mathcal{X}, d)$ and any spanning tree T of $\mathcal{X}$. Then the spanning tree $\hat{T}$ produced by Algorithm 5 has degree at most 5, and satisfies $w(\hat{T}) \leq 2w(T)$. Moreover, the algorithm can be run in $O(n)$ time.*

Proof The runtime of the algorithm is straightforward, so we analyze the other two claims. For any vertex u, let $\pi(u)$ be its parent in T. We first observe that the tree $\hat{T}$ produced has degree at most 5. To see this, fix any node u, and note that edges are added adjacent to u only when the for loop is called on u and on the parent $\pi(u)$. In the case it is called on u, at most two children (out-edges) are added to u., and when called on the parent, again at most two children and one parent (in-edge) are added to u. Interpreting the in-edge as a parent and out-edges as children, we have that $\hat{T}$ is a rooted tree with the same root as T, where each node has at most 4 children. We can then define $\hat{\pi}(u)$ to be the parent of u in $\hat{T}$.

Next, we analyze the cost of the tree. Observe that $w_d(T) = \sum_{u \in X} d(u, \pi(u))$. Thus it suffices to show that

$$d(u, \hat{\pi}(u)) \leq 2d(u, \pi(u)) \tag{2}$$

for any non-root node u. To see this, note that the parent $\hat{\pi}(u)$ is set when the for loop is called on the parent $v = \pi(u)$ of u. In this case, we order the children $C_v = \{x_1, \ldots, x_k\}$ so that $d(v, x_1) \leq d(v, x_2) \leq \cdots \leq d(v, x_k)$, where $u = x_i$ for some $i \in \{1, \ldots, k\}$. First note that if $k \leq 2$, then we have $\hat{\pi}(u) = v = \pi(u)$, so (2) holds trivially. Otherwise, we have $\hat{\pi}(u) = x_j$ for some $j < i$ (interpreting $x_0 = v$). By the ordering, and employing the triangle inequality, we have

$$\begin{aligned} d(u, \hat{\pi}(u)) &= d(x_i, x_j) \\ &\leq d(x_i, v) + d(x_j, v) \\ &\leq 2d(x_i, v) \\ &= 2d(u, \pi(u)), \end{aligned} \tag{3}$$

which completes the proof. ■

E.1. Analysis of the Algorithm

To begin, it will simplify the presentation if we can assume that all distances are unique, which we do now.

Fact E.1 *Let $\mathcal{A}$ be any algorithm that satisfies the correctness guarantees of Theorem 4 under the assumption that the set of distance $\{d(x, y)\}_{(x,y)\in\binom{\mathcal{X}}{2}}$ are unique. Then there is an algorithm $\mathcal{A}'$ that satisfies the correctness guarantees of Theorem 4 without this assumption.*

Proof The proof is by modifying the input to $\mathcal{A}$ to satisfy this. Specifically, we for every (x, y), we replace the input $\tilde{d}(x, y)$ with $\tilde{d}(x, y) + \varepsilon_{x,y}$, where $\varepsilon_{x,y} \sim [\varepsilon/2, \varepsilon]$ is i.i.d. and uniform for an arbitrarily small value of ϵ. It is easy to verify that the result is still a metric, as we always have $r_{x,y} \leq \varepsilon \leq r_{x,z} + r_{z,y}$ for any (x, y, z). We run $\mathcal{A}$ on the modified distances $\tilde{d}(x, y) + r_{x,y}$, which we can interpret as coming from the modified original metric with distances $d(x, y) + r_{x,y}$. Since $\{d(x, y)\}_{(x,y)\in\binom{\mathcal{X}}{2}}$ are now unique, the correctness of $\mathcal{A}$ follows. Moreover, since we have changed each distance in $d(x, y)$ by at most $\varepsilon << 1/\text{poly}(n) \cdot \min_{x,y} d(x, y)$, it follows that the cost of any spanning tree is changed by at most a $(1 - 1/\text{poly}(n))$ factor, which completes the proof. ■

Given Fact E.1, in what follows we will always assume that all distances are unique. We can now introduce the notion of a ℓ-heavy ball, whose definition relies on this fact.

Definition 23 (ℓ-heavy ball) *Fix any $\ell \geq 0$. For any point $v \in \mathcal{X}$, we define the level-ℓ heavy radius at the point v to be the smallest radius $r = r_v^\ell$ such that the metric ball $\mathcal{B}_d^\ell(v, r)$ under the distance measure d contains exactly 2^ℓ points in $\mathcal{X}$. We define the level-ℓ heavy ball at v to be the metric-ball $\mathcal{B}_d^\ell(v, r_v^\ell)$ under the metric d.*

Note that the existence of a radius r_v^ℓ such that $\mathcal{B}_d^\ell(v, r)$ contains exactly 2^ℓ points is guaranteed by the uniqueness of distances. We use the notion of ℓ-th level heavy balls in Definition 23 to show the following probabilistic guarantee for distance corruptions under the metric constraints.

Lemma 24 (Probabilistic metric violation guarantee) *Let $\widetilde{d}$ be the corrupted distance of d that satisfies the metric property. Fix any three points $x, y, u \in V$ and level $\ell \geq 2$, such that $x \in \mathcal{B}_d^\ell(u, r_u^\ell)$. Then with probability at least $1 - 2^{-c_\delta \cdot 2^\ell}$, where c_δ is a constant depending only on the corruption probability δ, the following inequalities hold:*

$$\widetilde{d}(x, y) \leq d(x, y) + 4 \cdot r_u^\ell;$$

$$d(x, y) \leq \widetilde{d}(x, y) + 4 \cdot r_u^\ell.$$

Proof For any point $z \in \mathcal{B}^\ell(u, r_u^\ell) \setminus \{x, y\}$, let $\mathcal{E}_z$ be the event that both $\widetilde{d}(y, z) = d(y, z)$ and $\widetilde{d}(x, z) = d(x, z)$; i.e., neither pair is corrupted. Then the probability that no such $\mathcal{E}_z$ holds for any z is at most

$$\Pr\left(\bigcap_{z \in \mathcal{B}^\ell(u, r_u^\ell) \setminus \{x,y\}} \neg \mathcal{E}_z\right) \le (1 - (1-\delta)^2)^{2^{\ell-1}} \le 2^{-c_\delta \cdot 2^\ell},$$

for some c_δ which is a constant depending only on δ. Thus, with probability at least $1 - 2^{-c_\delta \cdot 2^\ell}$, there exists at least one such $z \in \mathcal{B}^\ell(u, r_u^\ell) \setminus \{x, y\}$ with $\widetilde{d}(y, z) = d(y, z)$ and $\widetilde{d}(x, z) = d(x, z)$. Condition on this event, and fix it z now. We have

$$\begin{aligned}
\widetilde{d}(x, y) &\le \widetilde{d}(x, z) + \widetilde{d}(z, y) && \text{(triangle inequaliy for } \widetilde{d}) \\
&= d(x, z) + d(z, y) && \text{(the event } \mathcal{E}_z) \\
&\le 2d(x, z) + d(x, y) && \text{(triangle inequality for } d) \\
&\le d(x, y) + 4 \cdot r_u^\ell. && (x, z \in \mathcal{B}^\ell(u, r_u^\ell))
\end{aligned}$$

Similarly, for the second inequality, we have

$$\begin{aligned}
d(x, y) &\le d(x, z) + d(z, y) && \text{(triangle inequaliy for } d) \\
&= d(x, z) + \widetilde{d}(z, y) && \text{(the event } \mathcal{E}_z) \\
&\le 2r_u^\ell + \widetilde{d}(x, z) + \widetilde{d}(x, y) && (d(x, z) \le r_v \text{ and triangle inequality for } \widetilde{d}) \\
&= 2r_u^\ell + d(x, z) + \widetilde{d}(x, y) && \text{(the event } \mathcal{E}_z) \\
&\le \widetilde{d}(x, y) + 4 \cdot r_u^\ell, && (d(x, z) \le r_v)
\end{aligned}$$

as desired. ■

We now use Lemma 24 to prove the approximation guarantee for the MST. We first define a partition of the metric space $\mathcal{X}$ via a ball-carving. Note that the following procedure is not algorithmic, and is only used in the analysis. Also, in what follows, recall that we define T_d^* and $T_{\widetilde{d}}^*$ be the MST under the metric d and $\widetilde{d}$ respectively.

Algorithm 6 Level-ℓ Heavy Ball Carving

Input: Set of points V, integer $\ell \ge 1$

Output: Set of metric balls $\mathcal{B}_1, \ldots, \mathcal{B}_k$ covering V, and partition $S_1, \ldots, S_k$ of V such that $S_i \subseteq \mathcal{B}_i$ for all $i \in [k]$.

Initialize $i = 1$ and $X = V$.

while $X \neq \emptyset$ **do**

  $x_i = \arg\max_{y \in X} r_y^\ell$.

  Set $r_i^\ell = r_v^\ell$, $\mathcal{B}_i = \mathcal{B}^\ell(x_i, r_i^\ell)$, and $S_i^\ell = \mathcal{B}_i \cap X$.

  $X \leftarrow X \setminus S_i^\ell$

  $i \leftarrow i + 1$

end

Claim E.2 *Fix any $1 \leq \ell \leq \log n$, and let T_d^* be the minimum spanning tree of $G = (V, E)$ under distance d. Then we have*

$$w_d(T_d^*) \geq \frac{1}{2} \cdot \sum_i r_i^\ell,$$

where $\{r_i^\ell\}$ are the radii of the level-ℓ ball-carving from Algorithm 6.

Proof To avoid redundancy of notation, we drop the superscript of ℓ since the proofs on every ℓ are the same. We claim that the balls of radius $\mathcal{B}(x_i, \frac{r_i}{2})$ are disjoint (where x_i is the selected center of the ball of the i-th iteration). To see this, note that for a fixed i, if $\mathcal{B}(x_i, \frac{r_i}{2})$ contains a point of $z \in \mathcal{B}(x_j, r_j)$ for $j > i$, then the ball $\mathcal{B}(x_i, r_i)$ should have contained x_j (since $r_j < r_i$), and x_j should not have been selected during the ball carving process, which forms a contradiction. Similarly, if $\mathcal{B}(x_i, \frac{r_i}{2})$ contains a point of $z \in \mathcal{B}(x_k, r_k)$ for $k < i$, then the ball $\mathcal{B}(x_k, r_k)$ should have contained x_i, and x_i should not have been selected during the ball carving process. Finally, since the points in each $\mathcal{B}(x_i, \frac{r_i}{2})$ are disjoint, the minimum spanning tree must pay a cost to travel from the boundary to the center of each disjoint ball, which pays at least $\frac{1}{2} \cdot \sum_i r_i$ in cost. ■

Besides lower-bounding the MST cost as in Claim E.2, we present two technical steps toward the proof of Theorem 4.

Lemma 25 *Let T be any spanning tree of the set of points $\mathcal{X}$. Then we have*

$$\mathbb{E}\left[w_{\widetilde{d}}(T)\right] \leq w_d(T) + O(1) \cdot w_d(T_d^*),$$

where the expectation is over the randomness over which distances are corrupted.

Proof We root the tree T arbitrarily and define the charging scheme as follows. First note, by Lemma 24, for any pair (x, y), the event in Lemma 24 holds with probability $1 - 1/\mathrm{poly}(n)$ for at least one value of ℓ with $\ell = O(\log n)$. It follows by a union bound over $O(n^2)$ pairs that, with high probability, all pairs (x, y) satisfy the random event in Lemma 24 for some $\ell = O(\log n)$.

A Charging Scheme.
For each tree edge $(u, v) \in T$:

1. Suppose w.l.o.g. that v is the child node. Let ℓ be the smallest integer such that the ball $\mathcal{B}^\ell(x_i, r_i^\ell)$ satisfies the following properties:
    - v is included in $\mathcal{B}^\ell(x_i, r_i^\ell)$; and
    - $\widetilde{d}(u, v) \leq d(u, v) + 4 \cdot r_i^\ell$.
2. Distribute a charge of $4 \cdot r_i^\ell$ to $\mathcal{B}^\ell(x_i, r_i^\ell)$

It is straightforward that $\sum_{(u,v)\in T} \tilde{d}(u, v) \leq \sum_{(u,v)\in T} \tilde{d}(u, v) + C$, where C is the sum of all charges distributed to balls in the above process. Thus it suffices to upper bound C. To this end, we bound the expected number of times for a given ball at level ℓ to be charged. Note that such a ball $\mathcal{B}^\ell(x_i, r_i^\ell)$ can be charged at most once for each of the at most 2^ℓ points x contained in that ball. Moreover, to be charged by the point v, it must be that we did not have $\widetilde{d}(u,v) \le d(u,v) + 4 \cdot r_i^{\ell-1}$, which occurs with probaiblity at most $2^{-c_\delta 2^{\ell-1}}$ by Lemma 24. Define X_i^ℓ to be the number of times that $\mathcal{B}^\ell(x_i, r_i^\ell)$ is charged. Then, letting $X^\ell(p)$ be the event that ℓ is the level at which the point p is charged in the above scheme, we have

$$\begin{aligned}
\mathbb{E}\left[X_i^\ell\right] &\le \sum_{v \in \mathcal{B}^\ell(x_i, r_i^\ell)} \mathbb{E}\left[X^\ell(v)\right] \\
&\le \sum_{v \in \mathcal{B}^\ell(x_i, r_i^\ell)} 2^{-c_\delta 2^{\ell-1}} \\
&\le 2^\ell 2^{-c_\delta 2^{\ell-1}} \le \frac{c'_\delta}{2^\ell}
\end{aligned}$$

for some $c'_\delta \ge 0$ which is another constant. Thus, we can bound the total cost of the charging scheme by

$$\sum_{\ell=1}^{\log n} \sum_i \mathbb{E}\left[4 r_i^\ell X_i^\ell\right] \le \sum_{\ell=1}^{\log n} \sum_i 4 \frac{c'_\delta}{2^\ell} \cdot r_i^\ell \le \sum_{\ell=1}^{\log n} \frac{8}{2^\ell} c'_\delta w_d(T_d^*) = 16 c'_\delta w_d(T_d^*),$$

where the last inequality follows from Claim E.2 and the fact that the sum is geometric. Putting these bounds together, we have

$$\begin{aligned}
\mathbb{E}\left[w_{\widetilde{d}}(T)\right] &\le \sum_{(u,v)\in T} d(u,v) + \sum_{\ell=1}^{\log n} \sum_i \mathbb{E}\left[4 r_i^\ell X_i^\ell\right] \\
&\le w_d(T) + O(1) \cdot w_d(T_d^*).
\end{aligned}$$

as desired. ■

Lower Bounding $\tilde{w}(T)$ for any tree T We now prove an inequality in the reverse direction, demonstrating that, with high probability over the choice of corrupted distances, for any spanning tree T the cost of $\tilde{w}(T)$ is not too small. We begin with the following definition. In what follows, we fix $\alpha = \Theta(\sqrt{\log n})$ with a sufficiently large constant. For any ℓ, we will write $\mathcal{B}_1^\ell, \mathcal{B}_2^\ell, \ldots, \mathcal{B}_k^\ell$ to denote the set of balls produced by the Level-ℓ Heavy Ball Carving (Algorithm 6), and let r_i^ℓ denote the radius of $\mathcal{B}_i^\ell$. Note that by construction, each ball $\mathcal{B}_i^\ell$ contains exactly 2^ℓ points. In what follows, set $\beta = 1 - \delta$, and note that $\beta = \Omega(1)$ is at least a constant.

Definition 26 *Fix α as above, and fix any $x \in \mathcal{X}$. Let $\mathcal{B}_i^\ell$ be the ball in the level-ℓ heavy ball carving containing x, where $\ell = \log(\alpha)$. Then we say that x is good if at least $\beta 2^{\ell-1}$ distances in the set $\{(x,y) \mid y \in \mathcal{B}_i^\ell\}$ are not corrupted. Call a point bad if it is not good.*

Proposition 27 *With probability $1 - 1/\mathrm{poly}(n)$, the following holds: for every pair of two good points (x,y), we have*

$$d(x,y) \le \tilde{d}(x,y) + 4(r_i^\ell + r_j^\ell)$$

where (i,j) is such that $x \in \mathcal{B}_i^\ell, y \in \mathcal{B}_j^\ell$, and $\ell = \log \alpha$.

Proof We prove that for any two balls $\mathcal{B}_i^\ell, \mathcal{B}_j^\ell$ (possibly with $i = j$), and any subsets $S_i \subset \mathcal{B}_i^\ell, S_j \subset \mathcal{B}_j^\ell$ with size $|S_i|, |S_j| \geq \beta 2^{\ell-1}$, there exists at least one $w \in S_i, z \in S_j$ such that $(w, z) \notin \textsc{Corrupt}$. Let $\mathcal{E}$ denote this event, we prove that $\Pr[\mathcal{E}] \geq 1 - 1/\text{poly}(n)$. Given any two sets S_i, S_j, there are at least $\binom{\beta 2^{\ell-1}}{2} = s = \Theta(\alpha^2) > 100 \log n/\beta$ distinct pairs $(w, z) \in S_i \times S_j$ (where we used that α is set with a sufficiently large constant depending on $1/\beta$). Since each pair is corrupted independently with probability δ, the probability that all pairs $(w, z) \in S_i \times S_j$ are contained in $\textsc{Corrupt}$ is at most

$$(\delta)^{\frac{100 \log n}{(1-\delta)}} = (1 - (1-\delta))^{\frac{100 \log n}{(1-\delta)}} \leq \left(\frac{1}{2}\right)^{100 \log n} = n^{-100},$$

where the inequality follows using the fact that $(1-x)^r \leq \frac{1}{1+rx}$ for all $x \in [-\frac{1}{r}, 1]$ and $r \geq 0$. We can then union bound over all $\binom{n}{2} 2^\alpha = O(n^2) \cdot 2^{O(\sqrt{logn})} < O(n^3)$ such choices of S_i, S_j to obtain the desired result with probability at least $1 - n^{-95}$.

Now conditioned on $\mathcal{E}$, we can fix any two points x, y as in the statement of the proposition, where $x \in \mathcal{B}_i^\ell, y \in \mathcal{B}_j^\ell$. Since x, y are both good, there exist the desired sets $S_i \subset \mathcal{B}_i^\ell, S_j \subset \mathcal{B}_j^\ell$ with size $|S_i|, |S_j| \geq \beta 2^{\ell-1}$, such that all pairs $(x, u) \in S_i$ and $(y, v) \in S_j$ are not corrupted. By $\mathcal{E}$, we can then fix $w \in S_i$ and $z \in S_j$ such that $(w, z) \notin \textsc{Corrupt}$ is also not corrupted. We then have

$$\begin{aligned} d(x,y) &\leq d(x,w) + d(w,z) + d(z,y) \\ &\leq 2r_i^\ell + \tilde{d}(w,z) + 2r_j^\ell \\ &\leq 2(r_i^\ell + r_j^\ell) + \tilde{d}(w,x) + \tilde{d}(x,y) + \tilde{d}(y,z) \\ &= 2(r_i^\ell + r_j^\ell) + d(w,x) + \tilde{d}(x,y) + d(y,z) \\ &= 4(r_i^\ell + r_j^\ell) + \tilde{d}(x,y), \end{aligned} \tag{4}$$

which completes the proof. ∎

We now consider how to bound the cost of edges (x, y) where at least one of x, y is bad. To do so, set ℓ^* such that $2^{\ell^*} = c^* \log n$ with a large enough constant c^*, and consider the level-ℓ^* heavy ball carving $\mathcal{B}_1^{\ell^*}, \mathcal{B}_2^{\ell^*}, \ldots, \mathcal{B}_t^{\ell^*}$. Recall that each such ball has exactly $c^* \log n$ points. We have the following.

Fact E.3 *With probability at least $1 - 1/\text{poly}(n)$, for every $i \in [t]$, the ball $\mathcal{B}_i^{\ell^*}$ contains at most $\sqrt{\log n}$ bad points.*

Proof Note that for any point $x \in \mathcal{B}_i^{\ell^*}$, we have

$$\mathbb{E}\left[\left|\{(x,y) \mid y \in \mathcal{B}_i^{\ell^*}\} \cap \textsc{Corrupt}\right|\right] = \delta 2^{\ell^*}$$

And recall that x is bad if $\left|\{(x,y) \mid y \in \mathcal{B}_i^{\ell^*}\} \cap \textsc{Corrupt}\right| < \delta 2^{\ell^*-1}$. Thus, by Chernoff bounds, a point is bad with probability at most $2^{-\Theta(\alpha)} < 2^{-100\sqrt{\log n}}$. Thus, the probability that any fixed set S of $\sqrt{\log n}$ points is simultaniously bad is at most $(2^{-100\sqrt{\log n}})^{\sqrt{\log n}} =$

$1/n^{100}$. It follows that the probability that any set of more than $\sqrt{\log n}$ points in $\mathcal{B}_i^{\ell^*}$ is bad is at most

$$\begin{aligned}\Pr\left[B_i^{\ell^*} \text{ contains at least } \sqrt{\log n} \text{ bad points }\right] &\leq \binom{c^* \log n}{\sqrt{\log n}} \cdot n^{-100} \\ &\leq (c^* \log n)^{\sqrt{\log n}} n^{-100} \\ &\leq n^{-99}\end{aligned} \tag{5}$$

Union bounding over $t \leq n$ possible balls yields the desired result. ■

Fact E.4 *With probability at least $1 - 1/\mathrm{poly}(n)$, the following holds: for every pair $(x, y) \in \mathcal{X}$, where $x \in \mathcal{B}_i^{\ell^*}$, we have*

$$d(x, y) \leq \tilde{d}(x, y) + 4r_i^{\ell^*}.$$

Proof Note that for any x, with probability at least $(1 - (1 - \delta))^{c^* \log n} > 1 - n^{-100}$, there exists at least one $z \in \mathcal{B}_i^{\ell^*}$ with $(x, z) \notin$ Corrupt and $(z, y) \notin$ Corrupt. Conditioned on this, we have:

$$\begin{aligned}d(x, y) &\leq d(x, z) + d(z, y) \\ &\leq 2r_i^{\ell^*} + \tilde{d}(z, y) \\ &\leq 2r_i^{\ell^*} + \tilde{d}(z, x) + \tilde{d}(x, y) \\ &\leq 4r_i^{\ell^*} + \tilde{d}(x, y).\end{aligned} \tag{6}$$

The fact follows after union bounding over $O(n^2)$ pairs (x, y). ■

Proposition 28 *With probability at least $1 - 1/\mathrm{poly}(n)$, the following holds: for every spanning tree T of $\mathcal{X}$ with degree at most Δ, we have*

$$w(T) \leq \tilde{w}(T) + O(\Delta\sqrt{\log n}) \min_{T'} w(T').$$

Proof We first condition on the events in Proposition 27, and Facts E.3 and E.4, which all occur with probability $1 - 1/\mathrm{poly}(n)$. For any edge $(x, y) \in T$, if both x, y are good, we have

$$d(x, y) \leq \tilde{d}(x, y) + 4(r_i^{\ell} + r_j^{\ell}), \tag{7}$$

where $\ell = \log(\alpha)$, and $x \in \mathcal{B}_i^{\ell}, y \in \mathcal{B}_j^{\ell}$. Otherwise, if at least one of x, y is bad, then fix one of the points which is bad, w.l.o.g. we fix x which is bad. Then by E.4, we have

$$d(x, y) \leq \tilde{d}(x, y) + 4r_{\tau}^{\ell^*}), \tag{8}$$

where $x \in \mathcal{B}_{\tau}^{\ell^*}$. Now to bound the cost $\sum_{(x,y)\in T} d(x, y)$, we will bound each summation by either equation 7 or 8, depending on whether both x, y are good or if at least one is bad. We now bound the total cost of doing so.

Using fact E.3, we know that each ball $\mathcal{B}_\tau^{\ell^*}$ has at most $\sqrt{\log n}$ bad points points. Moreover, this ball can only contribute a cost of $4r_\tau^{\ell^*}$ at most $O(\Delta)$ times for each bad point in $\mathcal{B}_\tau^{\ell^*}$. Thus, over all edges $(x, y) \in T$, the term $4r_\tau^{\ell^*}$ appears on the RHS of the above equation at most $O(\Delta\sqrt{\log n})$ times. Similarly, for ball $\mathcal{B}_j^\ell$ at level $\ell = \log \alpha$, the term r_i^ℓ can only appear in the RHS of equation 7 when considering an edge with at least one endpoint in $\mathcal{B}_j^\ell$. Since $|\mathcal{B}_j^\ell| < O(\sqrt{\log n})$, again this radius is counted at most $O(\Delta\sqrt{\log n})$ times. It follows that

$$\begin{aligned}\sum_{(x,y)\in T} d(x,y) &\leq \sum_{(x,y)\in T} \tilde{d}(x,y) + O(\Delta\sqrt{\log n})\left(\sum_i r_i^\ell + \sum_j r_j^{\ell^*}\right) \\ &\leq \sum_{(x,y)\in T} \tilde{d}(x,y) + O(\Delta\sqrt{\log n}) \min_{T'} w(T'),\end{aligned} \tag{9}$$

as needed, where we used Claim E.2 in the last inequality. ■

We are now ready to prove Theorem 4.

Proof of Theorem 4 Let $T^* = \arg\min_T w(T)$. Letting $\tilde{T} = \arg\min_T \tilde{w}(T)$, we then set the output of our algorithm to be the result $\hat{T}$ of running Algorithm 5 on $\tilde{T}$ in the corrupted metric $\tilde{d}$. By proposition 22, the tree $\hat{T}$ has degree at most 5 and $\tilde{w}(\hat{T}) \leq 2\tilde{w}(\tilde{T})$. Then By Proposition 28, we have

$$\begin{aligned}\mathbb{E}\left[w(\hat{T})\right] &\leq \mathbb{E}\left[\tilde{w}(\hat{T})\right] + O(\sqrt{\log n})w(T^*) \\ &\leq 2\mathbb{E}\left[\tilde{w}(\tilde{T})\right] + O(\sqrt{\log n})w(T^*) \\ &\leq 2\mathbb{E}\left[\tilde{w}(T^*)\right] + O(\sqrt{\log n})w(T^*) \\ &\leq 2w(T^*) + O(1)\cdot w(T^*) + O(\sqrt{\log n})w(T^*) \\ &= O(\sqrt{\log n})w(T^*),\end{aligned} \tag{10}$$

where in the first line we applied Proposition 28, the second line used that $\hat{T}$ was a 2-approximation of the optimal MST in the corrupted space $\tilde{d}$, the third line used that $\tilde{T}$ is optimal for $\tilde{d}$, and in the fourth line we applied Lemma 25. ■

Appendix F. Lower Bounds

We give lower bounds for k-clustering and MST in this section. In particular, we show that

- For k-clustering, we show that any k-center, k-means, or k-median algorithm that provides *bounded* approximation under the weak-strong oracle model requires $\Omega(k)$ strong (point) oracle queries.
- For metric MST, we show that if we want to go below the approximation factor of $\sqrt{\log n}$, we have to make $\tilde{\Omega}(n)$ strong (point) oracle queries.
- Finally, for non-metric MST, we show that with $o(n)$ strong (point) oracle queries, we cannot break an approximation lower bound of $\Omega(\log n)$.

Our lower bound demonstrates that the algorithms we designed in Appendix B, Sections 3 and 4 are nearly tight up to $\mathrm{polylog}\, n$ factors, and one could not hope for algorithms that are significantly more efficient. Furthermore, by our lower bound on non-metric MST, we separate the complexity between the metric and non-metric cases.

F.1. Lower Bounds for k-Clustering

In the prior sections, we provided $O(1)$-approximation algorithms for k-center, k-means and k-median clustering, each using $\tilde{O}(k)$ queries to the strong oracle. A natural question is whether the strong location oracle is even necessary for these tasks: namely, is it possible to obtain a good approximation with the weak oracle alone? We demonstrate that this is not possible in a strong sense. Namely, we prove that $\Omega(k)$-strong oracle queries are necessary for any algorithm that achieves *any* bounded approximation for k-clustering tasks. Our main result is as follows:

Theorem 29 *Fix any positive real number $c \in \mathbb{R}^+$, and positive integer k larger than some constant, and fix the corruption probability to be $\delta = 1/3$. Then any algorithm* ALG *which produces a solution for either k-centers, k-means, or k-medians that, with probability at least $1/2$, has cost at most $c \cdot \mathsf{OPT}$, (where $\mathsf{OPT}(d)$ is the optimal solution to the clustering task in question) must make at least $\Omega(k)$ queries to a strong point oracle, or at least $\Omega(k^2)$ queries to a strong edge oracle.*

Proof We focus on the proof of the k-center case, and the lower bounds for k-means and k-median clustering follow the same construction. The construction of the hard distribution over inputs is as follows. The distribution will be over distances d for a fixed set of n points $\mathcal{X}$. We first assume that k is odd, and later generalize to the case of even k. Moreover, since any s-query point strong oracle algorithm implies a s^2-query edge strong oracle algorithm, it suffices to prove a $\Omega(k^2)$-query lower bound against edge strong oracles, since this will imply a $\Omega(k)$ lower bound for point strong oracle queries.

Construction of the ground-truth metric $(\mathcal{X}, d)$.

1. Partition $\mathcal{X}$ into sets S, O, so that S has the first $|S| = \frac{3}{2}(k-1)$ points (under some fixed ordering), and O has all remaining points.
2. Select a uniformly random subset $N \subset S$ of exactly $k-1$ points, and define $U = S \setminus N$.
3. Fix a uniformly random perfect matching M over N, so that $|M| = \frac{k-1}{2}$.
4. Define the metric d as follows:

$$d(x, y) = \begin{cases} 1 & \text{if } (x, y) \in M \\ 1 & \text{if } (x, y) \in O \times O \\ c & \text{otherwise} \end{cases}$$

It is straightforward to verify that the construction of d is a metric. We now describe how to generate the corrupted distances $\tilde{d}$ for a given draw of d. Specifically, the weak oracle will corrupt *at most one* distance $d(x, y)$.

Observe that the optimal k-centers clustering of the original metric $(\mathcal{X}, d)$ has a cost of 1, and must have exactly one center in O, one center chosen from each of the matched pairs $(x, y) \in M$, and one center placed at every unmatched point $y \in U$. Note that if a single one of these clusters does not have a center placed in it, the cost of the solution is at least c.

Construction of the weak-oracle metric $\tilde{d}$

1. Fix an arbitrary pair $(x^*, y^*) \in M$ such that $(x^*, y^*) \in$ Corrupt is corrupted. If no such pair exists, set $\tilde{d} = d$, otherwise:
2. Set $\tilde{d}(x^*, y^*) = c$, and for all other pairs $(x, y) \in \binom{n}{2} \setminus \{(x^*, y^*)\}$, set $\tilde{d}(x, y) = d(x, y)$.

Let $\mathcal{E}_1$ be the event that at least one pair $(x, y) \in M$ exists such that $(x, y) \in$ Corrupt. Note that $\Pr(\mathcal{E}_1) > 1 - (2/3)^{(k-1)/2} = 1 - 2^{-\frac{k-1}{4}}$. We now condition on this holding. Now consider the metric $\tilde{d}$ produced by the weak oracle conditioned on $\mathcal{E}_1$. Now consider the distances $\tilde{d}$ produced by the weak oracle. They consists of the cluster O of points pairwise distance 1 apart within O, and distance c away from all points not in O. It also consists of the matching $M' = M \setminus (x^*, y^*)$, where $\tilde{d}(x, y) = 1$ for all $(x, y) \in M'$, and then it consists of the $k/2$ points $S \cup \{x^*, y^*\}$ which are each distance c from all other points in $\mathcal{X}$. Notice, however, that the pair x^*, y^* that was corrupted was not known to the algorithm. Moreover, since N was chosen uniformly at random, and the matching M was uniformly random, if we let T be the set of $|T| = k/2$ points in $\tilde{d}$ that are distance c from all other points, it follows that the identity of the corrupted distance (x^*, y^*) is a uniformly random pair chosen from T.

Now consider any sequence $s_1, s_2, \ldots, \in \binom{\mathcal{X}}{2}$ of adaptive, possible randomized edge strong oracle queries made by an algorithm. Since $d(x, y) = \tilde{d}(x, y)$ for all pairs x, y such that at least one of $x, y \notin T$, we can assume WLOG that each $s_i \in \binom{T}{2}$ (otherwise it reveals no information to the algorithm). Now for any prefix $s_1, \ldots, s_i$, condition on the event $\mathcal{Q}_i$ that $s_1 \neq (x^*, y^*), s_2 \neq (x^*, y^*), \ldots s_i \neq (x^*, y^*)$. Conditioned on $\mathcal{Q}_i$, it still holds that the single corrupted pair (x^*, y^*) is still uniformly distributed over the set $\binom{T}{2} \setminus \{s_1, \ldots, s_i\}$. Thus, for any s_{i+1} with $i + 1 < k^2/100$, we have

$$\begin{aligned} \Pr(s_{i+1} \neq (x^*, y^*) \mid \mathcal{Q}_i) &= \Pr(\mathcal{Q}_{i+1} \mid \mathcal{Q}_i) \\ &= 1 - \frac{1}{\binom{|T|}{2} - i} \\ &> 1 - \frac{16}{k^2} \end{aligned} \tag{11}$$

Thus, if the algorithm makes a total of $\ell < k^2/1600$ strong edge oracle queries, we have

$$\Pr(\mathcal{Q}_\ell) > \left(1 - \frac{16}{k^2}\right)^\ell > 24/25$$

It follows that, conditioned on $\mathcal{E}_1$, with probability at least $24/25$, the algorithm does not find the corrupted pair. Condition on this event $\mathcal{Q}_\ell$ now, and condition on any output clustering $\mathcal{C}$ of the algorithm given the observations $s_1, \dots, s_\ell$. First, suppose that $\mathcal{C}$ does not contain exactly $k/2-1$ clusters in the set T. If it contains more, then either it does not contain a center in O, or it does not contain a center in a matching $(x, y) \in M'$, in either case it pays a cost of c. Thus, we can assume it has exactly $k/2-1$ clusters in T. Let $z \in T$ be the one point in T not opened as a center. If $z \notin \{x^*, y^*\}$, then clear ALG pays a k-centers cost of c. We show this happens with good probability.

To see this, note that since the oracle queried at most $\ell < k^2/1600$ points, it follows that the corrupted distance (x^*, y^*) is still uniformly distributed over the $\binom{k/2}{2} - \ell > k^2/32$ distances not queried within $T \times T$. Since at most $k/2$ of those distances can include z, the probability that $z \in \{x^*, y^*\}$ is at most $\frac{16}{k}$, in which case the algorithm pays a c approximation. Thus, conditioned on $\mathcal{E}_1, \mathcal{Q}_\ell$, the algorithm ALG still pays a c approximation with probaiblity at least $\frac{16}{k}$. Thus, by a union bound, ALG pays a c-approximation with probability at least $1 - (\frac{1}{25} + 2^{-\frac{k-1}{4}} + \frac{16}{k}) > 1/2$, which completes the proof for odd k.

Lastly, to handle the case when k is even, we can use the same instance, except take a final point w^* from O and make it distance c^2 from all other points in both d and $\tilde{d}$ – a center must be placed at w^*, and the remaining problem is reduced to the above instance with $k-1$ centers (which is now odd). Finally, while the above lower bound was for k-centers, note that the same instance implies a $\Omega(c/n)$ approximation lower bound against algorithms with the same query complexity for either k-means or k-median for algorithms. Since c can be made arbitrarily large, the result for k-means and k-medians follows. ■

F.2. Lower Bounds for Minimum Spanning Trees

In this section, we prove lower bounds for both the metric and non-metric MST problems.

F.2.1. Lower bound for Metric Minimum Spanning Tree.

We now prove a matching lower bound for the metric MST problem. Our construction is based on a instance with $O(n/\sqrt{\log n})$ well-seperated clusters $\{C_i\}_i$. We show that, with good probability, we can match nearly all clusters into pairs (C_i, C_j) such that *all* distances between C_i, C_j are corrupted. By corrupting these distances it will be impossible to recover the original clusters, which we show implies a $\Omega(\sqrt{\log n})$ approximation.

Theorem 30 *There exists a constant c such that any algorithm which outputs a spanning tree T of $(\mathcal{X}, d)$ such that $\mathbb{E}[w(T)] \leq c\sqrt{\log n} \cdot \min_{T'} w(T')$ in the weak-strong oracle model, must make at least $\Omega(n/\sqrt{\log n})$ strong oracle point queries. Moreover, this holds even when the weak-oracle distances $\tilde{d} : \mathcal{X}^2 \to \mathbb{R}$ is restricted to being a metric, and when the corruption probability is $\delta = 1/3$.*

To prove Theorem 30, we use the following standard result on large size matching in random graphs. We also provide a proof for completeness.

Fact F.1 *Let $G = (V, E)$ be a random graph where each edge (i, j) exists independently with probability at least $\rho > c \log n/n$, for a sufficiently large constant c. Then with probability $1 - 1/\mathrm{poly}(n)$, there exists a matching $M \subset \binom{n}{2}$ in G with size at least $|M| > n/4$.*

Proof The proof is a simple application of the principle of deferred decisions. Order the vertices arbitrarily $x_1, x_2, , \ldots, x_n$. Let $Z_{i,j}$ be an indicator random variable for the event that $(x_i, x_j) \in E$. We build a set of matched points $M \subset [n]$. Initially, M is empty. We will walk through the points x_i for $i = 1, 2, \ldots, (3/4)n$, and show that each can be matched to a vertex if it was not previously matched already.

We first condition on the event $\mathcal{E}_\infty$ that $Z_{1,j}$ exists for at least one $x_j \notin M$, which occurs with high probability by a Chernoff bound. Fix that x_j to match to x_1, and add both x_j, x_1 to M. Now for $i = 2, \ldots, 3n/4$, either x_i is matched by step i, or we have that $\sum_{j>i, x_j \notin M} Z_{i,j}$ is a sum of i.i.d. indiacator variables with expectation at least $\rho(3n/4 - M)$. So if $|M| > n/2$, then we are done, otherwise $\mathbb{E}\left[\sum_{j>i, x_j \notin M} Z_{i,j}\right] > (c/4) \log n/n$. Thus, again by Chernoff bounds, with high probability there exists at least one $j > i$ with $x_j \notin M$ such that (x_i, x_j) is an edge, and we can match (x_i, x_j) and continue. Since each vertex x_i is matched with high probability for $i = 1, 2, \ldots, (3/4)n$, the fact follows from a union bound. ■

We now present the main lower bound of Theorem 30, for which we will employ the following input distribution over $(\mathcal{X}, d, \tilde{d})$.

The Hard Instance for MST

1. Set $k = \Theta(\sqrt{\log n})$, and draw a uniformly random mapping $f : [n] \to [n/k]$ conditioned on $|f^{-1}(j)| = k$ for all $i \in [n/k]$. Define the i-th block $B_i = f^{-1}(i)$, and $B = \{B_1, \ldots, B_{n/k}\}$.

2. Define the true distances as follows: we set $d(x, y) = 1$ for any pair $x, y \in B_i$ that are in the same block B_i for some i, and $d(x, y) = k$ otherwise.

3. Find a maximal matching $M \subset B \times B$ such that for all $(i, j) \in M$, and for all $x \in B_i, y \in B_j$ we have $(x, y) \in \textsc{Corrupt}$.

4. For all $(i, j) \in M$, and for all $x \in B_i, y \in B_j$ set the weak oracle distance to be $\tilde{d}(x, y) = 1$. For all other pairs (x', y'), set $\tilde{d}(x', y') = d(x', y')$

Proof of Theorem 30 First, note that it is easy to verify that the resulting corrupted distances $\tilde{d}$ are metric. This can be seen for the following reason: any set of distances $d' : \mathcal{X}^2 \to \mathbb{R}$ defined by a partition $P_1, \ldots, P_t$ of $[n]$ such that $d'(x, y) = 1$ for x, y in the same piece P_i of the partition, and $d'(x, y) = \ell$ otherwise, for some $\ell > 1$, is a metric. Finally, we note that that both $d, \tilde{d}$ are of this form.

We first prove the lower bound against algorithms that make *no* strong oracle queries. First, note that for any pair of blocks B_i, B_j, there are at most $k^2 < \log(n)/200$ pairs of distances $(x, y) \in B_i \times B_j$. The probability that all such pairs are corrupted is at most $(1/3)^{\log(n)/200} > n^{-1/100}$. Thus, by Claim F.1, with probability $1 - 1/\text{poly}(n)$ the matching M satisfies $|M| > \frac{n}{4k}$. Let $\mathcal{E}_1$ be the event that the matching is at least this large – we will now condition on $\mathcal{E}_1$ holding.

Now fix any draw of the corrupted distances $\tilde{d}$ observed by the algorithm. Also condition on the set of corrupted distances $\textsc{Corrupt}$, and the matching M — we will prove the lower bound even against an algorithm that is told the matching M over $B \times B$. Note that conditioning on $\tilde{d}, \textsc{Corrupt}, M$ does not determine the original metric d — specifically, the function f is not fully determined by $\tilde{d}, \textsc{Corrupt}, M$. Since an algorithm that makes no strong oracle queries sees only $\tilde{d}, M$, by Yao's min-max principle we can assume the algorithm is deterministic, and thus produces a tree T deterministically based on $\tilde{d}, M$ which, for the sake of contradiction, we suppose satisfies $\mathbb{E}[w(T)] \leq c'\sqrt{\log n} \cdot \min_{T'} w(T')$, where the expectation is taken over the remaining randomness in d after conditioning on $\tilde{d}, \textsc{Corrupt}, M$.

Now fix any arbitrary rooting of T, and let $\pi(u)$ be the parent of any vertex $u \in \mathcal{X}$ under this rooting. We will charge to each vertex $u \in \mathcal{X}$ the cost $d(u, \pi(u))$. We now condition on any set of identities of $B_j = f^{-1}(j) \subset [n]$ for every block B_j that is not matched under M. Additionally, for every matched pair of blocks B_i, B_j, we condition on the set of identities in the union $B_i \cup B_j$, but we *do not* condition on the individual sets B_i and B_j. Specifically, note that after conditioning on $B_i \cup B_j$ for any $x \in B_i \cup B_j$, we claim that $\Pr(f(x) = i) = \Pr(f(x) = j) = 1/2$. This holds because even conditioned on $\tilde{d}$, we have $\tilde{d}(x, y) = 1$ for ally $x, y \in B_i \cup B_j$, so shuffling the values of the identities in $B_i \cup B_j$ does not effect the observations of the algorithm.

Now consider any $u \in B_i$ such that $(B_i, B_j) \in M$ is matched. First, suppose that $\pi(u) \notin B_i \cup B_j$ – then $d(u, \pi(u)) = k$ for all possible realizations of the remaining randomness. If $\pi(u) \in B_i \cup B_j$, we claim that $d(u, \pi(u)) = k$ with probability $1/2$ over the remaining randomness in f. To see this, note that because f_i maps each point in $B_i \cup B_j$ to B_i or B_j uniformly at random (subject to the constraint that $|f^{-1}(j)| = |f^{-1}(j)| = k$). The constraint only makes it less likely that any pair $(u, \pi(u))$ are mapped to the same side, so:

$$\Pr(f(u, \pi(u)) = (i, j)) + \Pr(f(u, \pi(u)) = (j, i)) \geq \Pr(f(u, \pi(u)) = (i, i)) + \Pr(f(u, \pi(u)) = (j, j))$$

Moreover, whenever $f(u, \pi(u)) = (i, j)$, we have that $d(u, \pi(u)) = k$, which completes the claim. Given this, it follows that the expected value of $d(u, \pi(u))$ is at least $k/2$ for any u in a matched block B_i. Since $|M| > n/(4k)$, it follows that at least $n/2$ points are matched, and each has an edge to its parent in T with expected cost $k/2$, from which it follows that the expected cost of T is $\Omega(nk) = \Omega(n\sqrt{\log n})$. Since the true MST cost of $(\mathcal{X}, d)$ is always at most $O(n)$, resulting by creating a star on the set of points within each B_i, and then adding the edges for an arbitrary spanning tree with $n/k - 1$ vertices over the vertices $\{p_1, \ldots, p_{n/k}\}$, where $p_i \in B_i$ is an arbitrary representative vertex in B_i. This completes the proof of the lower bound against algorithms which make no strong oracle queries.

We now show how to generalize the above argument to algorithms that make at most $\frac{n}{100k}$ strong oracle queries. Similar to the above, we condition on the weak oracle mapping $\tilde{d}$ as well as the matching M. We now consider any set of $\frac{n}{100k}$ strong oracle queries made by the algorithm – let $S \subset [n]$ be the set of vertices queried. Since $|S| < \frac{n}{100k}$ and $|M| > \frac{n}{4k}$, it follows that there is a matching M' with $M' > \frac{n}{8k}$ such that for every $(B_i, B_j) \in M'$, we have $S \cap (B_i \cup B_j) = \emptyset$. It follows that, even after revealing the values of $d(x, y)$ for all $x, y \in S$, for every $x \in B_i \cup B_j$ where $(B_i, B_j) \in M'$, the function $f(x)$ is still uniformly distributed in $\{i, j\}$. The remainder of the arguement follows as above, with a loss of 2 in the expected cost of the algorithm attributed to the fact that we only have a matching of size $\frac{n}{8k}$ rather than $\frac{n}{4k}$. ■

Remark 31 *One may wonder whether we can extend the metric MST lower bound to strong oracle edge queries in the same manner of Theorem 29. Alas, with our analysis, we cannot get a lower bound as strong as $\tilde{\Omega}(n^2)$. By a simple black-box reduction, Theorem 30 implies a $\Omega(n/\sqrt{\log n})$ lower bound for strong oracle edge queries for any algorithm with $o(\sqrt{\log n})$ approximation. For estimating the* value *of the MST, this turns out to be (nearly) tight as there exists a $O(1)$ approximation with $\tilde{O}(n)$ queries by Czumaj and Sohler (2004). Exploring whether this is the case for constructing the actual MST with strong distance queries is an interesting direction to pursue.*

F.2.2. Lower Bounds for Non-Metric Minimum Spanning Tree.

We now consider the problem of computing an approximate MST in the general Weak-Strong Oracle model, where the corrupted weak-oracle distances $\tilde{d}$ is not necessarily a metric (i.e., $\tilde{d}$ can violate the triangle inequality). Whereas Theorem 4 demonstrates that a $O(\sqrt{\log n})$ approximation is possible in the metric-weak oracle case with *no* strong oracle queries. We now prove a $\Omega(\log n)$ approximation lower bound for any algorithm in the non-metric case, even if it makes $o(n/\log n)$ strong oracle queries, thereby strongly separating the two models.

Theorem 32 *There exists a constant c such that any algorithm which outputs a spanning tree T of $(\mathcal{X}, d)$ such that $\mathbb{E}[w(T)] \le c \log n \cdot \min_{T'} w(T')$ in the weak-strong oracle model (with corruption probability $\delta = 1/3$), must make at least $\Omega(n)$ queries to the strong oracle.*

Proof The construction of the hard distribution over inputs is as follows. The distribution will be over distances d for a fixed set of n points $\mathcal{X}$.

Construction of the ground-truth metric $(\mathcal{X}, d)$.

1. Set $k = \frac{\log n}{100}$, and draw a uniformly random mapping $f : [n] \to [n/k]$ conditioned on $|f^{-1}(j)| = k$ for all $i \in [n/k]$. Define the i-th block $B_i = f^{-1}(i)$, and $B = \{B_1, \dots, B_{n/k}\}$.
2. Define the metric d as follows:

$$d(x, y) = \begin{cases} 1 & \text{if } (x, y) \in B_i \text{ for some } i \in [n/k] \\ k & \text{otherwise} \end{cases}$$

It is straightforward to verify that the construction of d is a metric. We now describe how to generate the corrupted distances $\tilde{d}$ for a given draw of d.

Note that the optimal MST first conencts together all points within the same block (each of the $\Theta(n)$ edges paying a cost of 1 for each such edge), and then connects together the remaining n/k blocks, each with a cost of k. Thus $\min_T w(T) = \Theta(n)$.

Construction of the weak-oracle metric $\tilde{d}$

1. Define the random graph $H = (\mathcal{X}, \hat{E})$ as follow. We have $(x, y) \in E$, if and only if $x \in B_i, y \in B_j$ with $i \neq j$, and such that $(x, u) \in$ Corrupt and $(v, y) \in$ Corrupt for all $u \in B_j$ and $v \in B_i$.

2. Let $M \subset \mathcal{X} \times \mathcal{X}$ be a maximum matching in the graph H.
3. Define the weak-oracle output $\tilde{d}$ as follows: for every $(x, y) \in M$, where $x \in B_i, y \in B_j$, we set $\tilde{d}(x, u) = 1$ and $\tilde{d}(v, y) \in$ Corrupt for all $u \in B_j$ and $v \in B_i$. For all other pairs (x, y), we set $\tilde{d}(x, y) = d(x, y)$.

First note, by the same argument in the proof of Claim F.1, we will have that $|M| > n/4$ with high probability. Note that even though the setting is slightly different, because (x, y) can never be an edge if (x, y) are in the same block B_i, there are still at least $n - n/k$ possible edges which can be adjacent to any individual point x, and each exists with probability at least $(1/3)^k > 1/\sqrt{n}$ as needed for the proof of Claim F.1. Call the event that $|M| > n/4$ $\mathcal{E}_1$, and condition on it now.

We first prove the lower bound against algorithms that make no strong oracle queries. Now fix any draw of the observed distances $\tilde{d}$, and condition on the identities of the matching $M \in \mathcal{X} \times \mathcal{X}$. By a simple averaging argument (Yao's Min-max principle), if there was a randomized algorithm correct with probability at least $1/\text{poly}(n)$ against any given input, then there would be a deterministic algorithm correct against this input distribution with probability at least $1/\text{poly}(n)$. So given such an algorithm, after fixing $\tilde{d}$ we can fix the tree T output by the algorithm. Like in the proof of Theorem 30, we orient T arbitrarily, let $\pi(x)$ be the parent of x in T, and charge each vertex x with the cost $d(x, \pi(x))$.

Now note that for every pair $(x, y) \in M$, conditioned on the observations $\tilde{d}$ and matching M, for this match pair we have $\tilde{d}(x, u) = \tilde{d}(y, u)$ for all $u \in \mathcal{X}$. Thus, by the symmetry of the identities, for any fixed B_i, B_j such that $x \in B_i, y \in B_j$ occurs with non-zero probability over the remaining randomness, we have that $x \in B_i, y \in B_j$ occurs with the same probability as $x \in B_j, y \in B_i$. To see this formally, note that conditioned on any matching M, we can consturct a bijection between the remaining realizations of the randomness where $x \in B_i, y \in B_j$ and where $x \in B_j, y \in B_i$, simply by swapping the values of $f(x), f(y)$ – this is possible because, after conditioning on any set of values $\{f(z)\}_{z \in \mathcal{X} \setminus \{x,y\}}$, the marginals of $f(x)$ and $f(y)$ are identically distributed. Thus, for any fixed parent $\pi(x)$ of x, the probability that $\pi(x)$ is in the same block as x is at most $1/2$. Thus the expected cost $d(x, \pi(x))$ for any matched point x is at least $k/2$, since for any fixed block B_i containing $\pi(x)$, we have $x \notin \pi(x)$ with probaiblity at least $1/2$. Since there are $\Omega(n)$ matched points, it follows that the expected cost of the algorithm is at least $\Omega(nk) = \Omega(n \log n)$, which completes the proof for the case of algorithms which do not query the strong oracle.

Finally, for any algorithm that makes at most $n/100$ strong (point) oracle queries, notice that there are still $n/20$ pairs of matched (x, y) points such that neither were queried. For such pairs, the above claim still holds, namely that for any fixed B_i, B_j such that $x \in B_i, y \in B_j$ occurs with non-zero probability, both $(x \in B_i, y \in B_j)$ and $(x \in B_j, y \in B_i)$ occur with equal probability. Thus, for any parent $\pi(x)$, the point x will be in a different block from $\pi(x)$ with probability at least $1/2$, even conditioned on the strong oracle observations, the matching M, and $\tilde{d}$, and the rest of the proof proceeds as above.

Theorem 32 ■

Appendix G. Experiments

In this section, we experimentally validate the performances of our clustering algorithms. We compare our algorithms with benchmarks on two extremes: the "weak baseline", where the benchmark algorithm has access to only the WO queries, and the "strong baseline", where the benchmark algorithm has access to SO queries on the entire dataset. We demonstrate that:

(i) The weak baseline algorithms with only WO access produce very poor-quality solutions;

(ii) Our algorithm achieve costs that are competitive with the strong baseline that queries SO on the entire dataset, while only using SO queries on a very small fraction ($< 1\%$) of the points.

Datasets. As discussed in Section 1, our experiments use both synthetic data generated from the extensively-studied Stochastic Block Model (SBM) Holland et al. (1983); Dyer and Frieze (1989); Decelle et al. (2011); Abbe et al. (2015); Abbe and Sandon (2015); Hajek et al. (2016); Mossel et al. (2015) and the embeddings generated from the MNIST dataset with t-SNE and SVD Deng (2012); Van der Maaten and Hinton (2008). We construct the SBM model with $k = 7$ clusters: in the i-th cluster, we sample points from a Gaussian distribution $\mathcal{N}(\mu, I)$ with $\mu_i = 10^5$ and $\mu_j = 0$ for all $j \neq i$, and we use the ℓ_2 metric. As points sampled from the Gaussian distribution are concentrated, ground truth clusters are well-separated and the cost of misclustering even a single point is large. For the MNIST dataset, we run t-SNE and SVD embeddings with $60k$ training data, and embed into $d = 2$ dimensions for t-SNE and $d = 50$ for SVD.

In both scenarios, there are clear "ground truth" clusters for each point. As such, there is a natural weak-oracle corruption policy: for a pair of points (x_i, x_j), if x_i and x_j are in the same ground truth cluster, flip the distance to an arbitrary inter-cluster distance; otherwise, flip the distance to an arbitrary intra-cluster distance. For the synthetic dataset, this results in an SBM model.

Algorithm Implementations. We implement the weak and strong benchmarks with the farthest traversal algorithm Gonzalez (1985) for the k-center task and the celebrated k-means++ algorithm for the k-means task Arthur and Vassilvitskii (2007), and both are de facto choices in practice. To perform the Lloyds iteration for k-means, we reveal the embedding vectors on the points with the SO query to the algorithm. We also use k-means++ as the post-processing algorithm of the sampled set S in Algorithm 1[6]. The basic version of the experiments are carried out Macbook Pro with M1 chip and 16GB RAM. An optimized version for larger-scale datasets was run on an virtual compute cluster with 360GB RAM.

Figures and tables. We vary the parameters for sampling in our algorithms and obtain the curves for the clustering cost vs. number of strong oracle queries for different values of δ (the weak oracle corruption probability). For tables, in each setting of δ for different sampling parameters, we pick the run with the best query-cost trade-off by selecting the run that minimizes the value $|\mathsf{SO}| \cdot \mathsf{cost}^{10}$, where $|\mathsf{SO}|$ is the number of queries to the strong oracle. We do this in order to prevent selecting runs that make very few queries but have poor cost.

6. For the weak benchmark employing k-means++, we reveal all embeddings for the Lloyd iterations, which only helps that baseline. However, for our algorithms, we only reveal embeddings of points queried in S.

Table 2: The best query-cost trade-off point for the k-center and k-means algorithms on the SBM. 'Competitive ratio' means the ratio between the costs of our algorithms and the storng benchmark. The left column indicates the percentages of SO queries used.

	n	% of data queried for SO			Competitive ratio		
		$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$
k-center	10k	6.51	7.42	7.42	0.828	0.707	0.880
	20k	1.26	1.96	5.46	0.802	0.842	0.795
	50k	0.798	1.484	2.184	0.809	0.779	0.832
	100k	0.252	0.917	0.917	0.804	0.718	0.762
k-means	10k	5.55	3.51	13.19	1.089	1.053	1.175
	20k	1.975	1.78	8.57	1.216	1.086	1.191
	50k	1.038	0.82	2.342	1.142	1.062	1.125
	100k	0.555	0.44	1.31	1.141	1.218	1.25

SBM Experiments We test with corruption rate of $\delta = 0.1, 0.2$ and 0.3 with the scales of $n = 10k$, $20k$, $50k$, and $100k$. The cost (log scale) vs. strong oracle query curves and trade-off points for the k-center and k-means algorithms can be observed in Figures 2 and 3 and Table 2. In the plots of Figure 2, weak baseline and strong baseline are farthest traversal with access to only WO and SO on entire dataset respectively. In comparison, for the plots of Figure 3, the weak and strong baselines use k-means++ with zero SO queries and the entire set of SO queries, respectively.

As one would expect, in Figure 2, the k-center cost decreases drastically at some thhreshold (from $> 125k$ to ~ 7) since thereafter no point gets misclustered. Moreover, this threshold is quite small — the algorithm converges as early as the point where it queries SO for only $\sim 0.5\%$ of the total points. In contrast, the drop of cost for k-means as in Figure 3 demonstrate a more "smooth" manner. Nevertheless, both Figure 2 and Figure 3 show that the costs of k-clustering algorithms drop sharply and approach the optimal cost with a very low percentage of SO queries.

We then show in Table 2 the best query-cost trade-off points for the k-center and k-means algorithms. It can be observed that our algorithm consistently outperforms even the farthest traversal with SO queries on the entire dataset, while using queries only an extremely small fraction of the points. For the k-means experiments, our algorithm can provide a solution that is within a factor of $< 1.25\times$ of strong benchmark with SO queries on $0.5\% \sim 1.31\%$ of the points in the dataset. We can also observe that trend from Table 2 that the percentage of SO queries decreases as n becomes large, while the competitive ratio remains in the same range. When n is large (e.g., in the $100k$ case), outperforming the benchmark takes SO queries on only $< 1\%$ of the points.

MNIST Experiments: We now discuss the results on MNIST with t-SNE and SVD embeddings. It is well-known that the MNIST t-SNE embedding with $d = 2$ forms well-seperated clusters; however, the dichotomy between the distances of inter- and intra-cluster

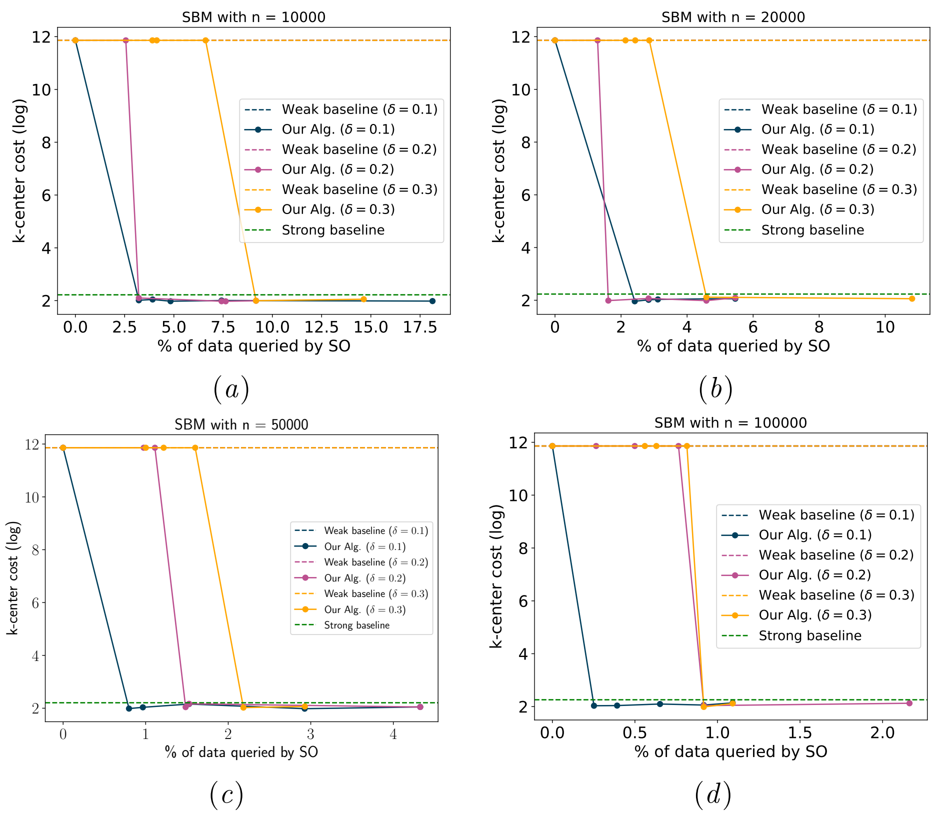


Figure 2: Number of SO queries vs. clustering cost under the SBM model for **k-center** with different values of δ and n. Weak baseline - farthest first traversal with WO queries only and Strong baseline - farthest first traversal with SO queries on full dataset.

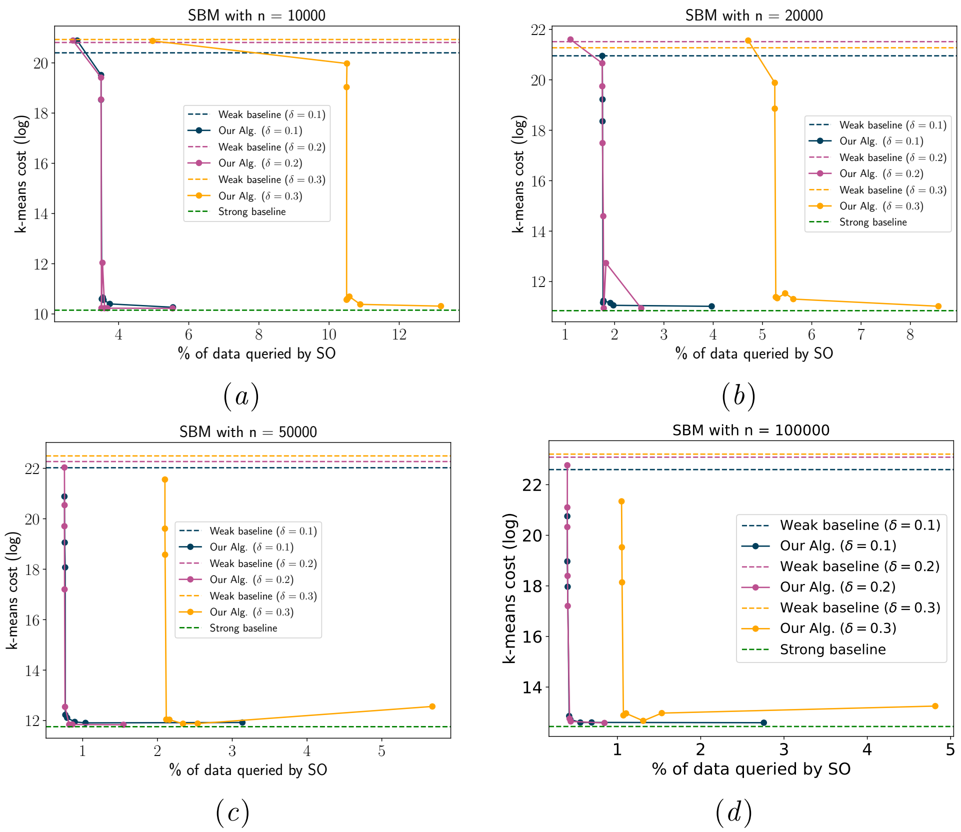
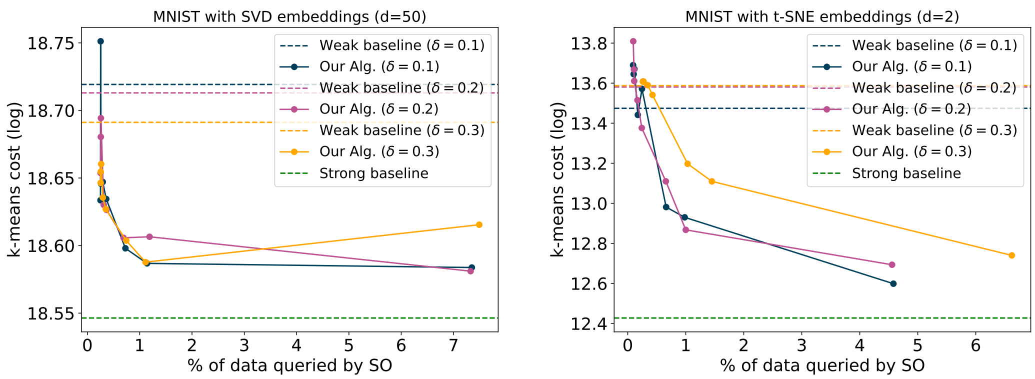


Figure 3: Number of SO queries vs. clustering cost under the SBM model for **k-means** with different values of δ and n. Weak baseline - k-means++ algorithm with WO queries only and Strong baseline - k-means++ algorithm with SO queries on full dataset.

points are not as stark as the SBM model. Furthermore, separations between clusters is notably worse for the SVD embedding. Thus, the t-SNE and SVD datasets are "less clustered" and "not clustered" instances, respectively. Figure 4 snd Table 3 show the query-cost curve for MNIST with t-SNE and SVD embeddings. Compared to the SBM model, the curves for these embeddings decrease less rapidly, a consequence of the clusters not being as well-separated. Nonetheless, our k-means algorithm still outperforms the weak benchmark by a significant margin using a small fraction of SO queries. For the t-SNE embeddings, as it is better-clustered, we observe a significant drop in cost after making less than 5% of the SO queries. On the other hand, for the not-well-clustered SVD embedding, although there is only a factor of ~ 1.2 between the weak and strong benchmark costs, our algorithm still manages to achieve non-trivial improvements in cost beyond the weak benchmark with fewer than 5% of the SO queries.

Table 3: The best query-cost trade-off point for the k-means algorithm on the MNIST embeddings.

	% of data queried for SO			Competitive ratio		
	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$	$\delta = 0.1$	$\delta = 0.2$	$\delta = 0.3$
t-SNE	4.58	4.57	6.62	1.169	1.286	1.367
SVD	0.25	0.311	0.253	1.121	1.109	1.105



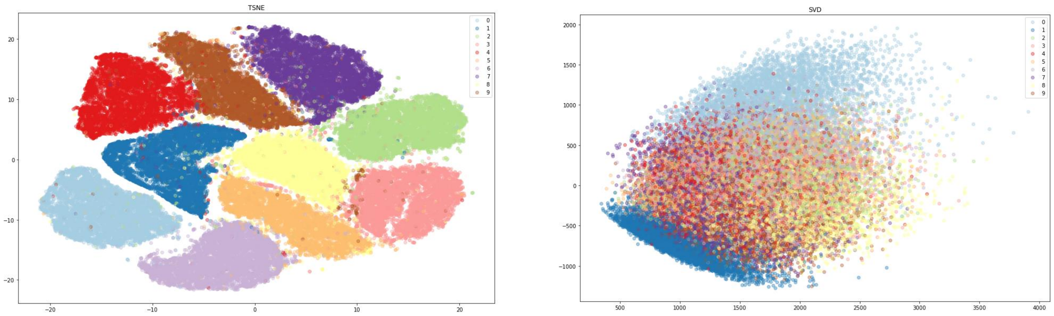
(a) (b)

Figure 4: Number of SO queries vs. k-means clustering cost under the **MNIST dataset** with different values of δ and n. Weak baseline - k-means++ algorithm with WO queries only and Strong baseline - k-means++ algorithm with SO queries on full dataset.

Appendix H. Ilustration of the clustering properties of MNIST tSNE and SVD embeddings

We have mentioned in Appendix G that there is a significant difference between the clustering costs for the MNIST embeddings obtained by tSNE and SVD methods. To give an intuitive justification of this statement, we include Figure 5 for the comparison of the embeddings. We note that these properties are well-known in the area, and we include them for completeness.

In Figure 5(a), the embeddings of different classes are well-clustered in general, although the distances between the clsuters are not as large as the instanced generated by the Stochastic Block Model. In contrast, in Figure 5(b), the separation between classes is unclear, and the costs generated by a "good" and a random clustering are comparable.



(a) tSNE (b) SVD

Figure 5: The comparison of the MNIST 60k embeddings plot with 2 dimensions. Left: the embeddings generated by tSNE, which are well-clsutered; Right: the embeddings generated by SVD, and we pick the first 2 dimensions for plot.