# Deterministic $k$-Median Clustering in Near-Optimal Time

Martín Costa*      Ermiya Farokhnejad†

## Abstract

The *metric $k$-median* problem is a textbook clustering problem. As input, we are given a metric space $V$ of size $n$ and an integer $k$, and our task is to find a subset $S \subseteq V$ of at most $k$ 'centers' that minimizes the total distance from each point in $V$ to its nearest center in $S$.

Mettu and Plaxton [UAI'02] gave a **randomized** algorithm for $k$-median that computes a $O(1)$-approximation in $\tilde{O}(nk)$ time.[1] They also showed that any algorithm for this problem with a bounded approximation ratio must have a running time of $\Omega(nk)$. Thus, the running time of their algorithm is optimal up to polylogarithmic factors.

For **deterministic** $k$-median, Guha et al. [FOCS'00] gave an algorithm that computes a poly$(\log(n/k))$-approximation in $\tilde{O}(nk)$ time, where the degree of the polynomial in the approximation is unspecified. To the best of our knowledge, this remains the state-of-the-art approximation of any deterministic $k$-median algorithm with this running time.

This leads us to the following natural question: **What is the best approximation of a deterministic $k$-median algorithm with near-optimal running time?** We make progress in answering this question by giving a deterministic algorithm that computes a $O(\log(n/k))$-approximation in $\tilde{O}(nk)$ time. We also provide a lower bound showing that any deterministic algorithm with this running time must have an approximation ratio of $\Omega(\log n/(\log k+\log \log n))$, *establishing a gap between the randomized and deterministic settings for $k$-median.*

## 1 Introduction

Clustering data is one of the fundamental tasks in unsupervised learning. As input, we are given a dataset, and our task is to partition the elements of the dataset into groups called *clusters* so that similar elements are placed in the same cluster and dissimilar elements are placed in different clusters. One of the basic formulations of clustering is *metric $k$-clustering*, where we are given a (weighted) metric space $(V, w, d)$ of size $n$, and our goal is to find a subset $S \subseteq V$ of at most $k$ *centers* that minimizes an objective function. We focus on the *$k$-median* problem, where the objective is defined as $\texttt{cost}(S) := \sum_{x \in V} w(x) \cdot d(x, S)$, where $d(x, S) := \min_{y \in S} d(x, y)$. Equivalently, we want to minimize the total weighted distance from points in $V$ to their nearest center in $S$.

**The State-of-the-Art for $k$-Median.** Metric $k$-median is a fundamental clustering problem with many real-world applications and has been studied extensively across many computational models [CGTS99, JV01, ANFSW19, BPR+17, COP03, AJM09]. The problem is NP-Hard and there is a long line of work designing efficient approximation algorithms for $k$-median using a variety of different techniques, such as local search [AGK+04] and Lagrangian relaxation [JV01]. Mettu and Plaxton gave a randomized algorithm for $k$-median that computes a $O(1)$-approximation in $\tilde{O}(nk)$ time [MP02], where the approximation guarantee holds with high probability. They also showed

---

*University of Warwick, `Martin.Costa@warwick.ac.uk`. Supported by a Google PhD Fellowship.

†University of Warwick, `Ermiya.Farokhnejad@warwick.ac.uk`

[1]We use $\tilde{O}(\cdot)$ to hide polylog factors in the size $n$ and the aspect ratio $\Delta$ (see Section 2) of the metric space.

that any algorithm for this problem with a non-trivial approximation ratio must have a running time of $\Omega(nk)$. It follows that their algorithm is *near-optimal*, i.e. optimal up to polylogarithmic factors in the running time and the constant in the approximation ratio.

**Deterministic Algorithms for $k$-Median.** Designing *deterministic* algorithms for fundamental problems is an important research direction within algorithms [CSS23, HLS24, ACS22, NS16, HLRW24]. Even though the randomized complexity of $k$-median is well understood, we do not have the same understanding of the problem in the deterministic setting. For deterministic $k$-median, Mettu and Plaxton gave an algorithm that computes a $O(1)$-approximation in $\tilde{O}(n^2)$ time [MP00], and Jain and Vazirani gave an algorithm with an improved approximation of 6 and a running time of $\tilde{O}(n^2)$ [JV01]. Whenever $k = \Omega(n)$, it follows from the lower bound of [MP02] that these algorithms are near-optimal in both the approximation ratio and running time. On the other hand, for $k \ll n$, these algorithms are slower than the randomized $O(1)$-approximation algorithm of [MP02]. Guha et al. gave an algorithm that computes a $\text{poly}(\log(n/k))$-approximation in a near-optimal running time of $\tilde{O}(nk)$ [GMMO00], where the degree of the polynomial in the approximation is unspecified. However, it is not clear how much this approximation ratio can be improved, and in particular, whether or not we can match the bounds for randomized algorithms. This leads us to the following question.

> **Question 1.** *What is the best approximation of any deterministic algorithm for $k$-median that runs in $\tilde{O}(nk)$ time?*

## 1.1 Our Results

We make progress in answering Question 1 by giving a deterministic algorithm with near-optimal running time and an improved approximation of $O(\log(n/k))$, proving the following theorem.

**Theorem 1.1.** *There is a deterministic algorithm for $k$-median that, given a metric space of size $n$, computes a $O(\log(n/k))$-approximate solution in $\tilde{O}(nk)$ time.*

We obtain our algorithm by adapting the "hierarchical partitioning" approach of Guha et al. [GMMO00]. We show that a modified version of this hierarchy can be implemented efficiently by using "restricted $k$-clustering" algorithms—a notation that was recently introduced by Bhattacharya et al. to design fast dynamic clustering algorithms [BCG$^+$24]. We design a deterministic algorithm for restricted $k$-median based on the reverse greedy algorithm of Chrobak et al. [CKY06] and combine it with the hierarchical partitioning framework to construct our algorithm.

In addition to our algorithm, we also provide a lower bound on the approximation ratio of any deterministic algorithm with a running time of $\tilde{O}(nk)$, proving the following theorem.

**Theorem 1.2.** *Any deterministic algorithm for $k$-median that runs in $\tilde{O}(nk)$ time when given a metric space of size $n$ has an approximation ratio of*

$$\Omega\left(\frac{\log n}{\log k + \log \log n}\right).$$

*This lower bound establishes a separation between the randomized and deterministic settings for $k$-median—ruling out the possibility of a deterministic $O(1)$-approximation algorithm that runs in near-optimal $\tilde{O}(nk)$ time for $k = n^{o(1)}$. For example, when $k = \text{poly}(\log n)$, Theorem 1.2 shows that any deterministic algorithm with a near-optimal running time must have an approximation ratio of*

2

$\Omega(\log n / \log \log n)$. On the other hand, Theorem 1.1 gives such an algorithm with an approximation ratio of $O(\log n)$, *which matches the lower bound up to a lower order $O(\log \log n)$ term.*

We prove Theorem 1.2 by adapting a lower bound on the *query complexity* of dynamic $k$-center given by Bateni et al. [BEF$^+$23], where the query complexity of an algorithm is the number of queries that it makes to the distance function $d(\cdot, \cdot)$. Our lower bound holds for any deterministic algorithm with a query complexity of $\tilde{O}(nk)$. Since the query complexity of an algorithm is a lower bound on its running time, this gives us Theorem 1.2. In general, establishing a gap between the deterministic and randomized query complexity of a problem is an interesting research direction [MN20]. Our lower bound implies such a gap for $k$-median when $k$ is sufficiently small.

For the special case of 1-median, Chang showed that, for any constant $\epsilon > 0$, any deterministic algorithm with a running time of $O(n^{1+\epsilon})$ has an approximation ratio of $\Omega(1/\epsilon)$ [Cha16]. The lower bound for 1-median by [Cha16] uses very similar techniques to the lower bounds of Bateni et al. [BEF$^+$23], which we adapt to obtain our result. In Theorem B.1, we provide a generalization of the lower bound in Theorem 1.2, giving a similar tradeoff between running time and approximation.

## 1.2 Related Work

Two other well-studied metric $k$-clustering problems are *k-means* and *k-center*. For $k$-means, the current state-of-the-art is the same as for $k$-median. Using randomization, it is known how to obtain a $O(1)$-approximation in $\tilde{O}(nk)$ time [MP02]. In Appendix A, we describe a generalization of the deterministic algorithm of [GMMO00] and show that it works for $k$-means as well as $k$-median. In particular, we can also obtain a $\text{poly}(\log(n/k))$-approximation for $k$-means in near-optimal $\tilde{O}(nk)$ time. On the other hand, for $k$-center, the situation is quite different. The classic greedy algorithm given by Gonzalez [Gon85] is deterministic and returns a 2-approximation in $O(nk)$ time. It is known that any non-trivial approximation algorithm must run in $\Omega(nk)$ time [BEF$^+$23], and it is NP-Hard to obtain a $(2 - \epsilon)$-approximation for any constant $\epsilon > 0$ [HN79]. Thus, this algorithm has exactly optimal approximation ratio and running time.

Another related line of work considers the specific case of Euclidean spaces. It was recently shown how to obtain a $\text{poly}(1/\epsilon)$-approximation in $\tilde{O}(n^{1+\epsilon+o(1)})$ time in such spaces [DS24]. They obtain their result by adapting the $\tilde{O}(n^2)$ time deterministic algorithm of [MP00] using locality-sensitive hashing.

The $k$-median problem has also recently received much attention in the *dynamic* setting, where points in the metric space are inserted and deleted over time and the objective is to maintain a good solution. A long line of work [CHP$^+$19, HK20, BCLP23, DHS24, BCG$^+$24, BCF25] recently led to a fully dynamic $k$-median algorithm with $O(1)$-approximation and $\tilde{O}(k)$ update time against adaptive adversaries, giving near-optimal update time and approximation.[2]

## 1.3 Organization

In Section 2, we give the preliminaries and describe the notation used throughout the paper. In Section 3, we give a technical overview of our results. We present our algorithm in Sections 4 and 5 and our lower bound in Appendix B. In Appendix C, we describe our results for the $k$-means problem.

---

[2]Note that, since we cannot obtain a running time of $o(nk)$ in the static setting, we cannot obtain an update time of $o(k)$ in the dynamic setting.

# 2   Preliminaries

Let $(V, w, d)$ be a weighted metric space of size $n$, where $w : V \longrightarrow \mathbb{R}_{\geq 0}$ is a weight function and $d : V \times V \longrightarrow \mathbb{R}_{\geq 0}$ is a metric satisfying the triangle inequality. The aspect ratio $\Delta$ of the metric space is the ratio of the maximum and minimum non-zero distances in the metric space. We use the notation $\tilde{O}(\cdot)$ to hide polylogarithmic factors in the size $n$ and the aspect ratio $\Delta$ of the metric space. Given subsets $S, U \subseteq V$, we define the cost of the solution $S$ with respect to $U$ as

$$\mathsf{cost}(S, U) := \sum_{x \in U} w(x) \cdot d(x, S),$$

where $d(x, S) := \min_{y \in S} d(x, y)$.[3] When we are considering the cost of $S$ w.r.t. the entire space $V$, we abbreviate $\mathsf{cost}(S, V)$ by $\mathsf{cost}(S)$. In the $k$-median problem on the metric space $(V, w, d)$, our objective is to find a subset $S \subseteq V$ of size at most $k$ which minimizes $\mathsf{cost}(S)$. Given an integer $k \geq 1$ and subsets $X, U \subseteq V$, we define the optimal cost of a solution of size $k$ within $X$ with respect to $U$ as

$$\mathrm{OPT}_k(U, X) := \min_{S \subseteq X, |S| = k} \mathsf{cost}(S, U).$$

When $X$ and $U$ are the same, we abbreviate $\mathrm{OPT}_k(U, X)$ by $\mathrm{OPT}_k(U)$. Thus, the optimal solution to the $k$-median problem on the metric space $(V, w, d)$ has cost $\mathrm{OPT}_k(V)$. For any $U \subseteq V$, we denote the metric subspace obtained by considering the metric $d$ and weights $w$ restricted to only the points in $U$ by $(U, w, d)$.

**The Projection Lemma.**   Given sets $A, B \subseteq V$, we let $\pi(A, B)$ denote the projection of $A$ onto the set $B$, which is defined as the subset of points $y \in B$ such that some point $x \in A$ has $y$ as its closest point in $B$ (breaking ties arbitrarily). In other words, we define $\pi(A, B) := \{\arg\min_{y \in B} d(y, x) \mid x \in A\}$. We use the following well-known *projection lemma* throughout the paper, which allows us to upper bound the cost of the projection $\pi(A, B)$ in terms of the costs of $A$ and $B$ [GT08, CKY06].

**Lemma 2.1.** *For any subsets $A, B \subseteq V$, we have that $\mathsf{cost}(\pi(A, B)) \leq \mathsf{cost}(B) + 2 \cdot \mathsf{cost}(A)$.*

*Proof.* Let $C$ denote $\pi(A, B)$. Let $x \in V$ and let $y^\star$ and $y$ be the closest points to $x$ in $A$ and $B$ respectively. Let $y'$ be the closest point to $y^\star$ in $C$. Then we have that

$$c(x, C) \leq d(x, y') \leq d(x, y^\star) + d(y^\star, y') \leq d(x, y^\star) + d(y^\star, y) \leq d(x, y) + 2 \cdot d(x, y^\star),$$

and so $d(x, C) \leq d(x, B) + 2 \cdot d(x, A)$. It follows that

$$\mathsf{cost}(C) = \sum_{x \in V} w(x) d(x, C) \leq \sum_{x \in V} w(x)(d(x, B) + 2 \cdot d(x, A)) = \mathsf{cost}(B) + 2 \cdot \mathsf{cost}(A). \qquad \square$$

The following well-known corollary of the projection lemma shows that, for any set $U \subseteq V$, the optimal cost of the $k$-median problem in $(U, w, d)$ changes by at most a factor of 2 if we are allowed to place centers anywhere in $V$.

**Corollary 2.2.** *For any subset $U \subseteq V$, we have that $\mathrm{OPT}_k(U) \leq 2 \cdot \mathrm{OPT}_k(U, V)$.*

*Proof.* Let $S_V^\star$ be a subset of $V$ of size at most $k$ that minimizes $\mathsf{cost}(S_V^\star, U)$ and let $S_U^\star = \pi(S_V^\star, U)$. Then, for any $x \in U$, it follows from Lemma 2.1 that $d(x, S_U^\star) \leq d(x, U) + 2 \cdot d(x, S_V^\star) = 2 \cdot d(x, S_V^\star)$, which implies the corollary. $\qquad \square$

---

[3]Note that we do not necessarily require that $S$ is a subset of $U$.

# 3 Technical Overview

We begin by describing the hierarchical partitioning approach used by Guha et al. [GMMO00] to obtain a poly($\log(n/k)$)-approximation algorithm with near-optimal running time. We then discuss the limitations of this approach and describe how we overcome these limitations to obtain our result.

## 3.1 The Hierarchical Partitioning Framework

Guha et al. [GMMO00] showed how to combine an $\tilde{O}(n^2)$ time $k$-median algorithm with a simple hierarchical partitioning procedure in order to produce a faster algorithm—while incurring some loss in the approximation. Their approach is based on the following divide-and-conquer procedure:

1. Partition the metric space $(V, w, d)$ into $q$ metric subspaces $(V_1, w, d), \ldots, (V_q, w, d)$.
2. Solve the $k$-median problem on each subspace $(V_i, w, d)$ to obtain a solution $S_i \subseteq V_i$.
3. Combine the solutions $S_1, \ldots, S_q$ to get a solution $S$ for the original space $(V, w, d)$.

The main challenge in this framework is implementing Step 3—finding a good way to merge the solutions from the subspaces into a solution for the original space. To implement this step, they prove the following lemma, which, at a high level, shows how to use the solutions $S_i$ to construct a *sparsifier* for the metric space $(V, w, d)$ that is much smaller than the size of the space.

**Lemma 3.1** ([GMMO00]). *Suppose that each solution $S_i$ is a $\beta$-approximate solution to the $k$-median problem in $(V_i, w, d)$. Let $V' = \bigcup_i S_i$ and, for each $y \in S_i$, let $w'(y)$ denote the total weight of points in $V_i$ that are assigned to $y$ in the solution $S_i$. Then any $\alpha$-approximate solution $S$ to the $k$-median problem in the space $(V', w', d)$ is a $O(\alpha\beta)$-approximation in the space $(V, w, d)$.*

Using Lemma 3.1, we can compute a (weighted) subspace $(V', w', d)$ that has size only $\sum_i |S_i| = O(kq)$. Crucially, we have the guarantee that any good solution that we find within this subspace is also a good solution in the space $(V, w, d)$. Thus, we can then use the deterministic $O(1)$-approximate $\tilde{O}(n^2)$ time $k$-median algorithm of [MP00] to compute a solution $S$ for $(V', w', d)$ in $\tilde{O}(k^2 q^2)$ time.

**A 2-Level Hierarchy.** Suppose we run this divide-and-conquer framework for one step (i.e. without recursing on the subspaces $(V_i, w, d)$) and just compute the solutions $S_i$ for $(V_i, w, d)$ using the $\tilde{O}(n^2)$ time algorithm of [MP00]. It follows immediately from Lemma 3.1 that the approximation ratio of the final solution $S$ is $O(1)$. We can also observe that, up to polylogarithmic factors, the total time taken to compute the $S_i$ is $\simeq q \cdot (n/q)^2 = n^2/q$, since the size of each subspace is $O(n/q)$. Furthermore, the time taken to compute $S$ is $\tilde{O}(k^2 q^2)$. By taking $q = (n/k)^{2/3}$, we get that the total running time of the algorithm is $\tilde{O}(nk \cdot (n/k)^{1/3})$, giving a polynomial improvement in the running time for $k \ll n$.

**An $\ell$-Level Hierarchy.** Now, suppose we run this framework for $\ell$ steps. To balance the running time required to compute the solutions at each level of this divide-and-conquer procedure, we want to subdivide each metric subspace at depth $i$ in the recursion tree into (roughly) $q_i = (n/k)^{2^{-i}}$ further subspaces. Guha et al. show that the running time of this algorithm is $\tilde{O}(nk \cdot (n/k)^{2^{-\ell}})$. By Lemma 3.1, we can also see that the approximation ratio of the final solution $S$ is $2^{O(\ell)}$. Setting $\delta = (n/k)^{2^{-\ell}}$, we get the following theorem.

**Theorem 3.2.** *There is a deterministic algorithm for $k$-median that, given a metric space of size $n$, computes a poly($\log(n/k)/\log \delta$)-approximation in $\tilde{O}(nk\delta)$ time, for any $2 \leq \delta \leq n/k$.*

Setting $\delta = O(1)$, we get immediately the following corollary.

**Corollary 3.3.** *There is a deterministic algorithm for k-median that, given a metric space of size n, computes a* $\mathrm{poly}(\log(n/k))$*-approximation in* $\tilde{O}(nk)$ *time.*

We remark that the results in [GMMO00] are presented differently and only claim an approximation ratio of $\mathrm{poly}(\log n/\log \delta)$. In Appendix A, we describe a generalization of their algorithm, proving Theorem 3.2 and also showing that it extends to k-means.

## 3.2 The Barrier to Improving The Approximation

While the sparsification technique that is described in Lemma 3.1 allows us to obtain much faster algorithms by sparsifying our input in a hierarchical manner, this approach has one major drawback. Namely, the fact that sparsifying the input in this manner also leads to an approximation ratio that scales exponentially with the number of levels in the hierarchy. Unfortunately, this exponential growth seems unavoidable with this approach. This leads us to the following question.

> **Question 2.** *Is there a different way to combine the solutions* $S_1, \ldots, S_q$ *that does* **not** *lead to exponential deterioration in the approximation?*

## 3.3 Idea I: Sparsification via Restricted k-Median

Very recently, Bhattacharya et al. introduced the notion of "restricted k-clustering" in order to design efficient and consistent dynamic clustering algorithms [BCG+24]. The **restricted k-median** problem on the space $(V, w, d)$ is the same as the k-median problem, except that we are also given a subset $X \subseteq V$ and have the additional restriction that our solution $S$ must be a subset of $X$. Crucially, even though the algorithm can only place centers in the solution $S$ within $X$, it receives the entire space $V$ as input and computes the cost of the solution $S$ w.r.t. the entire space.

The restricted k-median problem allows us to take a different approach toward implementing Step 3 of the divide-and-conquer framework. Instead of compressing the entire space $(V, w, d)$ into only $O(kq)$ many weighted points, we restrict the output of the solution $S$ to these $O(kq)$ points but still consider the rest of the space while computing the cost of the set $S$. This can be seen as a less aggressive way of sparsifying the metric space, where we lose less information. It turns out that this approach allows us to produce solutions of higher quality, where the approximation scales linearly in the number of levels in the hierarchy instead of exponentially.

Efficiently implementing this new hierarchy is challenging since we need to design a fast deterministic algorithm for restricted k-median with the appropriate approximation guarantees. To illustrate our approach while avoiding this challenge, we first describe a simple version of our algorithm with improved approximation and near-optimal *query complexity*. We later show how to design such a restricted algorithm, allowing us to implement this algorithm efficiently.

## 3.4 Our Algorithm With Near-Optimal Query Complexity

Let $(V, w, d)$ be a metric space of size $n$ and $k \leq n$ be an integer. We define the value $\ell := \log_2(n/k)$, which we use to describe our algorithm. Our algorithm works in the following 2 *phases*.

**Phase I:** In the first phase of our algorithm, we construct a sequence of partitions $Q_0, \ldots, Q_\ell$ of the metric space $V$, such that the partition $Q_i$ is a *refinement* of the partition $Q_{i-1}$.[4] We let $Q_0 := \{V\}$.

---

[4]i.e. for each element $X \in Q_{i-1}$, there are elements $X_1, \ldots, X_q \in Q_i$ such that $X = X_1 \cup \cdots \cup X_q$.

Subsequently, for each $i = 1, \ldots, \ell$, we construct the partition $Q_i$ by arbitrarily partitioning each $X \in Q_{i-1}$ into subsets $X_1$ and $X_2$ of equal size and adding these subsets to $Q_i$.[5] For $X \in Q_{i-1}$, we define $\mathcal{P}(X) := \{X' \in Q_i \mid X' \subseteq X\}$.

**Phase II:** The second phase of our algorithm proceeds in *iterations*, where we use the partitions $\{Q_i\}_i$ to compute the solution in a bottom-up manner. Let $V_{\ell+1}$ denote the set of points $V$. For each $i = \ell, \ldots, 0$, our algorithm constructs $V_i$ as follows:

> For each $X \in Q_i$, let $S_X$ be the optimal solution to the $k$-median problem in the metric space $(X, w, d)$ such that $S_X \subseteq X \cap V_{i+1}$. Let $V_i := \bigcup_{X \in Q_i} S_X$.

**Output:** The set $V_0$ is the final output of our algorithm.

The following theorem summarizes the behaviour of this algorithm.

**Theorem 3.4.** *There is a deterministic algorithm for $k$-median that, given a metric space of size $n$, computes a $O(\log(n/k))$-approximation with $\tilde{O}(nk)$ queries (but exponential running time).*

We now sketch the proof of Theorem 3.4 by outlining the analysis of the approximation ratio and query complexity. The formal proof follows from the more general analysis in Section 5.2.

**Approximation Ratio**

To bound the cost of the solution $V_0$ returned by our algorithm, we first need to be able to relate the costs of a solution in the hierarchy to the costs of the solutions in the subsequent levels. Given any set $X$ within a partition $Q_i$, the following key claim establishes the relationship between the cost of the solution $S_X$ on the metric subspace $(X, w, d)$ and the costs of the solutions $\{S_{X'}\}_{X' \in \mathcal{P}(X)}$ on the metric subspaces $\{(X', w, d)\}_{X' \in \mathcal{P}(X)}$.

**Claim 3.5.** *For any set $X \in \bigcup_{i=0}^{\ell-1} Q_i$, we have that*

$$\mathsf{cost}(S_X, X) \leq \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(S_{X'}, X') + O(1) \cdot \mathrm{OPT}_k(X).$$

*Proof.* Let $R$ denote the set $\bigcup_{X' \in \mathcal{P}(X)} S_{X'}$. Let $S^\star$ be an optimal solution to the $k$-median problem in the metric space $(X, w, d)$ and let $S' := \pi(S^\star, R)$ be the projection of $S^\star$ onto $R$.

Since $S_X$ is the optimal solution to the $k$-median problem in $(X, w, d)$ such that $S_X \subseteq R$, it follows that $\mathsf{cost}(S_X, X) \leq \mathsf{cost}(S', X)$. Applying the projection lemma (Lemma 2.1) to the projection $S'$ of $S^\star$ onto $R$, we get that $\mathsf{cost}(S', X) \leq \mathsf{cost}(R, X) + 2 \cdot \mathsf{cost}(S^\star, X) = \mathsf{cost}(R, X) + O(1) \cdot \mathrm{OPT}_k(X)$. Combining these two inequalities, it follows that

$$\mathsf{cost}(S_X, X) \leq \mathsf{cost}(R, X) + O(1) \cdot \mathrm{OPT}_k(X). \tag{1}$$

The claim then follows from the fact that

$$\mathsf{cost}(R, X) = \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(R, X') \leq \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(S_{X'}, X'). \qquad \square$$

---

[5]Note that it might not be possible for the subsets $X_1$ and $X_2$ to have equal size. For simplicity, we ignore this detail in the technical overview.

By repeatedly applying Claim 3.5 to the sum $\sum_{X \in Q_i} \mathsf{cost}(S_X, X)$, we obtain the following upper bound on $\mathsf{cost}(V_0)$:

$$\mathsf{cost}(V_0) \leq \sum_{X \in Q_\ell} \mathsf{cost}(S_X, X) + O(\ell) \cdot \mathrm{OPT}_k(V). \tag{2}$$

We prove Equation (2) in Claim 5.3. Now, let $S^\star$ be an optimal solution to $k$-median in $(V, w, d)$ and consider any $X \in Q_\ell$. Since $S_X$ is an optimal solution to $k$-median in the subspace $(X, w, d)$, we have that $\mathsf{cost}(S_X, X) = \mathrm{OPT}_k(X) \leq 2 \cdot \mathrm{OPT}_k(X, V) \leq 2 \cdot \mathsf{cost}(S^\star, X)$, where the first inequality follows from Corollary 2.2. Thus, summing over each $X \in Q_\ell$, we get that

$$\sum_{X \in Q_\ell} \mathsf{cost}(S_X, X) \leq 2 \cdot \sum_{X \in Q_\ell} \mathsf{cost}(S^\star, X) = 2 \cdot \mathsf{cost}(S^\star) = 2 \cdot \mathrm{OPT}_k(V). \tag{3}$$

Combining Equations (2) and (3), it follows that $\mathsf{cost}(V_0) \leq O(\ell) \cdot \mathrm{OPT}_k(V)$. Thus, the solution $V_0$ returned by our algorithm is a $O(\ell) = O(\log(n/k))$ approximation.

**Query Complexity**

Since the partitions constructed in Phase I of the algorithm are arbitrary, we do not make any queries to the metric in Phase I. Thus, we now focus on bounding the query complexity of Phase II.

The total number of queries made during the $i^{th}$ iteration in Phase II is the sum of the number of queries required to compute the solutions $\{S_X\}_{X \in Q_i}$. In the first iteration (when $i = \ell$), we compute $|Q_\ell|$ many solutions, each one on a subspace of size $n/|Q_\ell|$.[6] We can trivially compute an optimal solution in $(X, w, d)$ by querying every distance between all $O(|X|^2)$ pairs of points in $X$ and then checking every possible solution. Thus, we can upper bound the number of queries by

$$\left(\frac{n}{|Q_\ell|}\right)^2 \cdot |Q_\ell| = \frac{n^2}{|Q_\ell|} = \frac{n^2}{2^\ell} = O(nk),$$

where we are using the fact that the number of sets in the partition $Q_\ell$ is $2^\ell = n/k$. For each subsequent iteration (when $0 \leq i < \ell$), we compute $|Q_i|$ many solutions, each one on a subspace $(X, w, d)$ of size at most $n/|Q_i|$, where the solution is restricted to the set $X \cap V_{i+1}$, which has size at most $|X \cap V_{i+1}| = |S_{X_1} \cup S_{X_2}| \leq 2k$, where $\mathcal{P}(X) = \{X_1, X_2\}$ and $S_{X_1}$ and $S_{X_2}$ are computed in the previous iteration. Since we only need to consider solutions that are contained in some subset of at most $O(k)$ points, we can find an optimal such restricted solution with $O(k) \cdot (n/|Q_i|)$ queries. It follows that the total number of queries that we make during this iteration is at most $O(nk/|Q_i|) \cdot |Q_i| \leq O(nk)$. Thus, the total query complexity of our algorithm is $\ell \cdot O(nk) = \tilde{O}(nk)$.

### 3.5  Idea II: A Deterministic Algorithm for Restricted $k$-Median

In order to establish the *approximation guarantees* of our algorithm in Section 3.4, we use the fact that, given a metric subspace $(V, w, d)$ of size $n$ and a subset $X \subseteq V$, we can find a solution $S \subseteq X$ to the $k$-median problem such that

$$\mathsf{cost}(S) \leq \mathsf{cost}(X) + O(1) \cdot \mathrm{OPT}_k(V). \tag{4}$$

We use this fact in the proof of Claim 3.5 (see Equation (1)), which is the key claim in our analysis of our algorithm. Furthermore, to establish the *query complexity* bound of our algorithm, we

---

[6]Again, we assume for simplicity that each set in the partition has the same size.

use the fact that we can find such a solution $S_X$ with $O(n|X|)$ many queries. To implement this algorithm efficiently, we need to be able to find such a solution in $\tilde{O}(n|X|)$ *time*. While designing an algorithm with these exact guarantees seems challenging (since it would require efficiently matching the bounds implied by the projection lemma), we can give an algorithm with the following relaxed guarantees, which suffice for our applications.

**Lemma 3.6.** *There is a deterministic algorithm that, given a metric space $(V, w, d)$ of size $n$, a subset $X \subseteq V$, and a parameter $k$, returns a solution $S \subseteq X$ of size $2k$ in $\tilde{O}(n|X|)$ time such that*

$$\texttt{cost}(S) \leq \texttt{cost}(X) + O\left(\log\left(\frac{|X|}{k}\right)\right) \cdot \mathrm{OPT}_k(V).$$

To prove Lemma 3.6, we give a simple modification of the reverse greedy algorithm of Chrobak, Kenyon, and Young [CKY06]. The reverse greedy algorithm initially creates a solution $S \leftarrow V$ consisting of the entire space, and proceeds to repeatedly peel off the point in $S$ whose removal leads to the smallest increase in the cost of $S$ until only $k$ points remain. Chrobak et al. showed that this algorithm has an approximation ratio of $O(\log n)$ and $\Omega(\log n / \log \log n)$. While this large approximation ratio might seem impractical for our purposes, we can make 2 simple modifications to the algorithm and analysis in order to obtain the guarantees in Lemma 3.6.

1. We start off by setting $S \leftarrow X$ instead of $S \leftarrow V$. This ensures that the output $S$ is a subset of $X$, and gives us the guarantee that $\texttt{cost}(S) \leq \texttt{cost}(X) + O(\log |X|) \cdot \mathrm{OPT}_k(V)$.

2. We return the set $S$ once $|S| \leq 2k$ instead of $|S| \leq k$. This allows us to obtain the guarantee that $\texttt{cost}(S) \leq \texttt{cost}(X) + O(\log(|X|/k)) \cdot \mathrm{OPT}_k(V)$.

We provide the formal proof of Lemma 3.6 in Section 4.

We can now make the following key observation: In our algorithm from Section 3.4, whenever we compute a solution $S_X$ to the $k$-median problem, we impose the constraint that $S_X \subseteq R$ for a set $R$ of size $R = O(k)$. Thus, if we use the algorithm from Lemma 3.6 to compute our solutions within the hierarchy, whenever we apply this lemma we can assume that $|X| = O(k)$. Consequently, the approximation guarantee that we get from Lemma 3.6 becomes

$$\texttt{cost}(S) \leq \texttt{cost}(X) + O(1) \cdot \mathrm{OPT}_k(V),$$

matching the required guarantee from Equation (4).

**Dealing with Bicriteria Approximations.** One caveat from Lemma 3.6 is that the solutions output by the algorithm have size $2k$ instead of $k$. In other words, these are "bicriteria approximations" to the $k$-median problem, i.e. solutions that contain more than $k$ points. Thus, the final output of our algorithm has size $2k$. By using the extraction technique of Guha et al. [GMMO00] described in Lemma 3.1, it is straightforward to compute a subset of $k$ of these points in $\tilde{O}(k^2)$ time while only incurring a $O(1)$ increase in the approximation ratio.

**Putting Everything Together.** By combining our algorithm from Section 3.4 with Lemma 3.6, we get our main result, which we prove in Section 5.

**Theorem 3.7.** *There is a deterministic algorithm for $k$-median that, given a metric space of size $n$, computes a $O(\log(n/k))$-approximate solution in $\tilde{O}(nk)$ time.*

### 3.6 Our Lower Bound for Deterministic $k$-Median

We also prove the following lower bound for deterministic $k$-median. Due to space constraints, we defer the proof of this theorem to Appendix B.

**Theorem 3.8.** *For every $\delta \geq 1$, any deterministic algorithm for the $k$-median problem that has a running time of $O(kn\delta)$ on a metric space of size $n$ has an approximation ratio of*

$$\Omega\left(\frac{\log n}{\log \log n + \log k + \log \delta}\right).$$

We prove this lower bound by adapting a lower bound construction by Bateni et al. [BEF$^+$23]. In this paper, the authors provide a lower bound for dynamic $k$-center clustering against adaptive adversaries. Although their primary focus is $k$-center, their lower bounds can be extended to various $k$-clustering problems. The main idea is to design an adaptive adversary that controls the underlying metric space as well as the points being inserted and deleted from the metric space. Whenever the algorithm queries the distance between any two points $x, y$, the adversary returns a value $d(x, y)$ that is consistent with the distances reported for all previously queried pairs of points. Note that if we have the distances between some points (not necessarily all of them), we might be able to embed different metrics on the current space that are consistent with the queried distances. More specifically, [BEF$^+$23] introduces two different consistent metrics, and shows that the algorithm can not distinguish between these metrics, leading to a large approximation ratio.

We use the same technique as described above with slight modifications. The first difference is that [BEF$^+$23] considers the problem in the fully dynamic setting, whereas our focus is on the static setting. The adversary has two advantages in the fully dynamic setting:

1. The adversary has the option to delete a (problematic) point from the space.
2. The approximation ratio of the algorithm is defined as the maximum approximation ratio throughout the entire stream of updates.

Both of these advantages are exploited in the framework of [BEF$^+$23]: The adversary removes any 'problematic' point $x$ and the approximation ratio of the algorithm is proven to be large only in special steps (referred to as 'clean operations') where the adversary has removed all of the problematic points. In the static setting, the adversary does not have these advantages, and the approximation ratio of the algorithm is only considered at the very end.

Due to the differences between the objective functions of the $k$-median and $k$-center problems, we observed that we can adapt the above framework for deterministic static $k$-median. One of the technical points here is to show that, if we have a problematic point $x$ that is contained in the output $S$ of the algorithm, we can construct the metric so that the cost of the cluster of $x$ in solution $S$ become large (see Claim B.3). The final metric space is similar to the 'uniform' metric introduced in [BEF$^+$23] with a small modification. Since the algorithm is **deterministic**, the output is the same set $S$ if we run the algorithm on this final metric again. Hence, we get our lower bounds for any deterministic algorithm for the static $k$-median problem.

## 4 A Deterministic Algorithm for Restricted $k$-Median

In this section, we prove the following theorem.

**Theorem 4.1.** *There is a deterministic algorithm that, given a metric space $(V, w, d)$ of size $n$, a subset $X \subseteq V$, and a parameter $k' \geq k$, returns a solution $S \subseteq X$ of size $k'$ in $\tilde{O}(n|X|)$ time such that*

$$\mathsf{cost}(S) \leq \mathsf{cost}(X) + O\left(\log\left(\frac{|X|}{k' - k + 1}\right)\right) \cdot \mathrm{OPT}_k(V).$$

This algorithm is based on a variant of the reverse greedy algorithm for the $k$-median problem [CKY06], which we modify for the restricted $k$-median problem. We refer to our modified version of this algorithm as RES-GREEDY$_{k'}$.

## 4.1 The Restricted Reverse Greedy Algorithm

Let $(V, w, d)$ be a metric space of size $n$, $X \subseteq V$ and $k' \leq |X|$ be an integer. The restricted reverse greedy algorithm RES-GREEDY$_{k'}$ begins by initializing a set $S \leftarrow X$ and does the following:

> While $|S| > k'$, identify the center $y \in S$ whose removal from $S$ causes the smallest increase in the cost of the solution $S$ (i.e. the point $y = \arg\min_{z \in S} \mathsf{cost}(S - z)$) and remove $y$ from $S$. Once $|S| \leq k'$, return the set $S$.

## 4.2 Analysis

Let $m$ denote the size of $X$ and let $k \leq k'$. Now, suppose that we run RES-GREEDY$_{k'}$ and consider the state of the set $S$ throughout the run of the algorithm. Since points are only removed from $S$, this gives us a sequence of nested subsets $S_{k'} \subseteq \cdots \subseteq S_m$, where $|S_i| = i$ for each $i \in [k', m]$. Note that $S_{k'}$ is the final output of our algorithm. The following lemma is the main technical lemma in the analysis of this algorithm.

**Lemma 4.2.** *For each $i \in [k' + 1, m]$, we have that*

$$\mathsf{cost}(S_{i-1}) - \mathsf{cost}(S_i) \leq \frac{2}{i - k} \cdot \mathrm{OPT}_k(V).$$

Using Lemma 4.2, we can now prove the desired approximation guarantee of our algorithm. By using a telescoping sum, we can express the cost of the solution $S_{k'}$ as

$$\mathsf{cost}(S_{k'}) = \mathsf{cost}(S_m) + \sum_{i=k'+1}^{m} \left(\mathsf{cost}(S_{i-1}) - \mathsf{cost}(S_i)\right). \tag{5}$$

Applying Lemma 4.2, we can upper bound the sum on the RHS of Equation (5) by

$$\sum_{i=k'+1}^{m} \left(\mathsf{cost}(S_{i-1}) - \mathsf{cost}(S_i)\right) \leq \sum_{i=k'+1}^{m} \frac{2}{i - k} \cdot \mathrm{OPT}_k(V) \leq O\left(\log\left(\frac{m}{k' - k + 1}\right)\right) \cdot \mathrm{OPT}_k(V). \tag{6}$$

Combining Equations (5) and (6), we get that

$$\mathsf{cost}(S_{k'}) \leq \mathsf{cost}(S_m) + O\left(\log\left(\frac{m}{k' - k + 1}\right)\right) \cdot \mathrm{OPT}_k(V),$$

giving us the desired bound in Theorem 4.1, since $S_m = X$ and $m = |X|$.

## 4.3 Proof of Lemma 4.2

The following analysis is the same as the analysis given in [CKY06], with some minor changes to accommodate the fact that we are using the algorithm for the restricted $k$-median problem and to allow us to obtain an improved approximation for $k' \geq (1 + \Omega(1))k$, at the cost of outputting bicriteria approximations.

We begin with the following claim, which we use later on in the analysis.

**Claim 4.3.** *For all subsets $A \subseteq B \subseteq V$, we have that*

$$\sum_{y \in B \setminus A} (\text{cost}(B - y) - \text{cost}(B)) \leq \text{cost}(A) - \text{cost}(B).$$

*Proof.* For each $y \in B$, let $C_B(y)$ denote the cluster of the points in $V$ that are assigned to $y$ within the solution $B$. In other words, $C_B(y)$ is the subset of points in $V$ whose closest point in $B$ is $y$ (breaking ties arbitrarily). We can now observe that

$$
\begin{aligned}
\sum_{y \in B \setminus A} (\text{cost}(B - y) - \text{cost}(B)) &= \sum_{y \in B \setminus A} \sum_{x \in V} w(x) \left( d(x, B - y) - d(x, B) \right) \\
&= \sum_{y \in B \setminus A} \sum_{x \in C_B(y)} w(x) \left( d(x, B - y) - d(x, B) \right) \\
&\leq \sum_{y \in B \setminus A} \sum_{x \in C_B(y)} w(x) \left( d(x, A) - d(x, B) \right) \\
&\leq \sum_{x \in V} w(x) \left( d(x, A) - d(x, B) \right) \\
&= \text{cost}(A) - \text{cost}(B).
\end{aligned}
$$

Here, the first and fifth lines follow from the definition of the cost function, the second line follows since $d(x, B - y) - d(x, B)$ can only be non-zero when $x \in C_B(y)$, the third line follows from the fact that $A \subseteq B - y$ for any $y \in B \setminus A$, and the fourth line from the fact that $\{C_B(y)\}_{y \in B}$ partition the set $V$. $\square$

Let $S^\star$ denote an optimal solution to the $k$-median problem in the metric space $(V, w, d)$ and let $i \in [k' + 1, m]$. We denote by $S_i'$ the projection $\pi(S^\star, S_i)$ of the optimal solution $S^\star$ onto the set $S_i$. It follows that

$$
\begin{aligned}
\text{cost}(S_{i-1}) - \text{cost}(S_i) &\leq \min_{y \in S_i \setminus S_i'} (\text{cost}(S_i - y) - \text{cost}(S_i)) \\
&\leq \frac{1}{|S_i \setminus S_i'|} \cdot \sum_{y \in S_i \setminus S_i'} (\text{cost}(S_i - y) - \text{cost}(S_i)) \\
&\leq \frac{1}{i - k} \cdot \sum_{y \in S_i \setminus S_i'} (\text{cost}(S_i - y) - \text{cost}(S_i)) \\
&\leq \frac{1}{i - k} \cdot \left( \text{cost}(S_i') - \text{cost}(S_i) \right) \\
&\leq \frac{2}{i - k} \cdot \text{cost}(S^\star) = \frac{2}{i - k} \cdot \text{OPT}_k(V).
\end{aligned}
$$

The first line follows directly from how the algorithm chooses which point to remove from $S_i$.[7] The second line follows from the fact that the minimum value within a set of real numbers is upper bounded by its average. The third line follows from the fact that $|S_i \setminus S_i'| \geq |S_i| - |S_i'| \geq i - k$. The fourth line follows from Claim 4.3. Finally, the fifth line follows from Lemma 2.1, which implies that $\mathtt{cost}(S_i') \leq \mathtt{cost}(S_i) + 2 \cdot \mathtt{cost}(S^\star)$.

## 4.4 Implementation

We now show how to implement RES-GREEDY to run in time $O(n|X|\log n)$. Our implementation uses similar data structures to the randomized local search in [BCG+24]. For each $x \in V$, we initialize a list $L_x$ which contains all of the points $y \in X$, sorted in increasing order according to the distances $d(x,y)$. We denote the $i^{th}$ point in the list $L_x$ by $L_x(i)$. We maintain the invariant that, at each point in time, each of the lists in $\mathcal{L} = \{L_x\}_{x \in V}$ contains exactly the points in $S$. Thus, at each point in time, we have that $\mathtt{cost}(S) = \sum_{x \in V} w(x)d(x, L_x(1))$. By implementing each of these lists using a balanced binary tree, we can initialize them in $O(n|X|\log n)$ time and update them in $O(n \log n)$ time after each removal of a point from $S$. Since the $S$ initially has size $|X|$, the total time spent updating the lists is $O(n|X|\log n)$. We also explicitly maintain the clustering $\mathcal{C} = \{C_S(y)\}_{y \in S}$ induced by the lists $\mathcal{L}$, where $C_S(y) := \{x \in V \mid L_x(1) = y\}$. We can initialize these clusters in $O(n)$ time given the collection of lists $\mathcal{L}$ and update them each time a list in $\mathcal{L}$ is updated while only incurring a $O(1)$ factor overhead in the running time. We now show that, using these data structures, we can implement each iteration of the greedy algorithm in $O(n)$ time. Since the algorithm runs for at most $|X|$ iterations, this gives us the desired running time.

**Implementing an Iteration of Greedy.** Using the lists $\mathcal{L}$ and clustering $\mathcal{C}$, we can compute

$$\mathtt{change}(y) \leftarrow \sum_{x \in C_S(y)} w(x)(d(x, L_x(2)) - d(x, L_x(1)))$$

for each $y \in S$. Since any point $x \in V$ appears in exactly one cluster in $\mathcal{C}$, this takes $O(n)$ time in total. By observing that removing $y$ from $S$ causes each point $x \in C_S(y)$ to be reassigned to the center $L_x(2)$, we can see that $\mathtt{change}(y)$ is precisely the value of $\mathtt{cost}(S - y) - \mathtt{cost}(S)$. Since minimizing $\mathtt{cost}(S - y) - \mathtt{cost}(S)$ is equivalent to minimizing $\mathtt{cost}(S - y)$, it follows that

$$\min_{z \in S} \mathtt{change}(z) = \min_{z \in S}(\mathtt{cost}(S - z) - \mathtt{cost}(S)) = \min_{z \in S} \mathtt{cost}(S - z).$$

Thus, we let $y \leftarrow \arg\min_{z \in S} \mathtt{change}(z)$, remove $y$ from $S$, and proceed to update the data structures. Excluding the time taken to update the data structures, the iteration takes $O(n)$ time.

## 5 Our Deterministic $k$-Median Algorithm

In this section, we prove Theorem 1.1, which we restate below.

**Theorem 5.1.** *There is a deterministic algorithm for $k$-median that, given a metric space of size $n$, computes a $O(\log(n/k))$-approximate solution in $\tilde{O}(nk)$ time.*

---

[7]Note that, for analytic purposes, we only take the minimum over $y \in S_i \setminus S_i'$ instead of all of $S_i$ in this inequality.

## 5.1 Our Algorithm

Let $(V, w, d)$ be a metric space of size $n$ and $k \leq n$ be an integer. We also define the value $\ell := \lceil \log_2(n/k) \rceil$, which we use to describe our algorithm. Our algorithm works in 3 *phases*, which we describe below.

**Phase I:** In the first phase of our algorithm, we construct a sequence of partitions $Q_0, \ldots, Q_\ell$ of the metric space $V$, such that the partition $Q_i$ is a *refinement* of the partition $Q_{i-1}$.[8] We start off by setting $Q_0 := \{V\}$. Subsequently, for each $i = 1, \ldots, \ell$, we construct the partition $Q_i$ as follows:

> Initialize $Q_i \leftarrow \varnothing$. Then, for each $X \in Q_{i-1}$, arbitrarily partition $X$ into subsets $X_1$ and $X_2$ such that $||X_1| - |X_2|| \leq 1$, and add these subsets to $Q_i$.

For $X \in Q_{i-1}$, we define $\mathcal{P}(X) := \{X' \in Q_i \mid X' \subseteq X\}$.

**Phase II:** The second phase of our algorithm proceeds in *iterations*, where we use the partitions $\{Q_i\}_i$ to compute the solution in a bottom-up manner. Let $V_{\ell+1}$ denote the set of points $V$. For each $i = \ell, \ldots, 0$, our algorithm constructs $V_i$ as follows:

> For each $X \in Q_i$, let $S_X$ be the solution obtained by running RES-GREEDY$_{2k}$ on the subspace $(X, w, d)$, restricting the output to be a subset of $X \cap V_{i+1}$. Finally, we define $V_i := \bigcup_{X \in Q_i} S_X$.

**Phase III:** Consider the set $V_0$ which contains $2k$ points and let $\sigma : V \longrightarrow V_0$ be the projection from $V$ to $V_0$. Define a weight function $w_0$ on each $y \in V_0$ by $w_0(y) := \sum_{x \in \sigma^{-1}(y)} w(x)$ (i.e. $w_0(y)$ is the total weight of all points in $V$ that are projected onto $y$). Let $S$ be the solution obtained by running the algorithm of Mettu-Plaxton [MP00] on the metric space $(V_0, w_0, d)$.

**Output:** The solution $S$ is the final output of our algorithm.

## 5.2 Analysis

We now analyze our algorithm by bounding its approximation ratio and running time.

**Approximation Ratio**

We begin by proving the following claim, which, for any set $X$ within a partition $Q_i$, allows us to express the cost of the solution $S_X$ w.r.t. the metric subspace $(X, w, d)$ in terms of the costs of the solutions $\{S_{X'}\}_{X' \in \mathcal{P}(X)}$.

**Claim 5.2.** *For any set $X \in \bigcup_{i=0}^{\ell-1} Q_i$, we have that*

$$\mathsf{cost}(S_X, X) \leq \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(S_{X'}, X') + O(1) \cdot \mathrm{OPT}_k(X).$$

---

[8]i.e. for each element $X \in Q_{i-1}$, there are elements $X_1, \ldots, X_q \in Q_i$ such that $X = X_1 \cup \cdots \cup X_q$.

*Proof.* Let $R$ denote the set $\bigcup_{X' \in \mathcal{P}(X)} S_{X'}$. We obtain the solution $S_X$ by calling RES-GREEDY$_{2k}$ on the metric space $(X, w, d)$ while restricting the output to be a subset of $R$. Thus, it follows from Theorem 4.1 that

$$\mathsf{cost}(S_X, X) \leq \mathsf{cost}(R, X) + O\left(\log\left(\frac{|R|}{2k - k + 1}\right)\right) \cdot \mathrm{OPT}_k(X).$$

By observing that $|R|/(k+1) \leq 4k/(k+1) \leq 4$, it follows that $\mathsf{cost}(S_X, X) \leq \mathsf{cost}(R, X) + O(1) \cdot \mathrm{OPT}_k(X)$. Finally, the claim follows since

$$\mathsf{cost}(R, X) = \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(R, X') \leq \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(S_{X'}, X'). \qquad \square$$

Using Claim 5.2, we now prove the following claim.

**Claim 5.3.** *For any $i \in [0, \ell]$, we have that*

$$\mathsf{cost}(V_0, V) \leq \sum_{X \in Q_i} \mathsf{cost}(S_X, X) + O(i) \cdot \mathrm{OPT}_k(V).$$

*Proof.* We prove this claim by induction. Note that the base case where $i = 0$ holds trivially. Now, suppose that the claim holds for some $i - 1 \in [0, \ell - 1]$. Then we have that

$$\mathsf{cost}(V_0, V) \leq \sum_{X \in Q_{i-1}} \mathsf{cost}(S_X, X) + O(i-1) \cdot \mathrm{OPT}_k(V)$$

$$\leq \sum_{X \in Q_{i-1}} \left( \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(S_{X'}, X') + O(1) \cdot \mathrm{OPT}_k(X) \right) + O(i-1) \cdot \mathrm{OPT}_k(V)$$

$$= \sum_{X \in Q_{i-1}} \sum_{X' \in \mathcal{P}(X)} \mathsf{cost}(S_{X'}, X') + O(1) \cdot \sum_{X \in Q_{i-1}} \mathrm{OPT}_k(X) + O(i-1) \cdot \mathrm{OPT}_k(V)$$

$$\leq \sum_{X \in Q_i} \mathsf{cost}(S_X, X) + O(1) \cdot \mathrm{OPT}_k(V) + O(i-1) \cdot \mathrm{OPT}_k(V)$$

$$= \sum_{X \in Q_i} \mathsf{cost}(S_X, X) + O(i) \cdot \mathrm{OPT}_k(V).$$

The second line follows from Claim 5.2. In the fourth line, we are using the fact that, for an optimal solution $S^\star$ of $V$ and any partition $Q$ of $V$, we have

$$\sum_{X \in Q} \mathrm{OPT}_k(X) \leq 2 \cdot \sum_{X \in Q} \mathrm{OPT}_k(X, V) \leq 2 \cdot \sum_{X \in Q} \mathsf{cost}(S^\star, X) = 2 \cdot \mathrm{OPT}_k(V),$$

where the first inequality follows from Corollary 2.2. $\qquad \square$

We get the following immediate corollary from Claim 5.3 by setting $i = \ell$.

**Corollary 5.4.** *We have that*

$$\mathsf{cost}(V_0, V) \leq \sum_{X \in Q_\ell} \mathsf{cost}(S_X, X) + O\left(\log\left(\frac{n}{k}\right)\right) \cdot \mathrm{OPT}_k(V).$$

Using Corollary 5.4, we prove the following lemma.

15

**Lemma 5.5.** *We have that* $\mathsf{cost}(V_0) = O(\log(n/k)) \cdot \mathrm{OPT}_k(V)$.

*Proof.* By Theorem 4.1, it follows that, for any $X \in Q_\ell$,

$$\mathsf{cost}(S_X, X) \leq \mathsf{cost}(X, X) + O\left(\log\left(\frac{|X|}{k}\right)\right) \cdot \mathrm{OPT}_k(X) \leq O\left(\log\left(\frac{n}{k}\right)\right) \cdot \mathrm{OPT}_k(X, V). \quad (7)$$

Combining Corollary 5.4 and Equation (7), we get that

$$\mathsf{cost}(V_0, V) \leq \sum_{X \in Q_\ell} \mathsf{cost}(S_X, X) + O\left(\log\left(\frac{n}{k}\right)\right) \cdot \mathrm{OPT}_k(V)$$

$$\leq O\left(\log\left(\frac{n}{k}\right)\right) \cdot \sum_{X \in Q_\ell} \mathrm{OPT}_k(X, V) + O\left(\log\left(\frac{n}{k}\right)\right) \cdot \mathrm{OPT}_k(V) \leq O\left(\log\left(\frac{n}{k}\right)\right) \cdot \mathrm{OPT}_k(V). \quad \square$$

By Lemma 5.5, we get that $V_0$ is a $O(\log(n/k))$-bicriteria approximation of size $2k$. Using the extraction technique of [GMMO00] (see Lemma 3.1 or Lemma A.5), which allows us to compute an exact solution to the $k$-median problem from a bicriteria approximation while only incurring constant loss in the approximation ratio, it follows that the solution $S$ constructed in Phase III is a $O(\log(n/k))$-approximation and has size at most $k$.

**Running Time**

We begin by proving the following lemma, which summarizes the relevant properties of the partitions constructed in Phase I of the algorithm.

**Lemma 5.6.** *For each* $i \in [0, \ell]$, *the set* $Q_i$ *is a partition of* $V$ *into* $2^i$ *many subsets of size at most* $n/|Q_i| + 2$.

*Proof.* We define $Q_0$ as $\{V\}$, which is a trivial partition of $V$. Now, suppose that this statement holds for the partition $Q_i$, where $0 \leq i < \ell$. The algorithm constructs $Q_{i+1}$ by taking each $X \in Q_i$ and further partitioning $X$ into subsets $X_1$ and $X_2$, such that difference in the sizes of these subsets is at most 1. We can also observe that the number of subsets in the partition $Q_{i+1}$ is $2 \cdot |Q_i| = 2 \cdot 2^i = 2^{i+1}$.[9] Since each subset $X \in Q_i$ has size at most $n/|Q_i| + 2$, it follows that each subset in $Q_{i+1}$ has size at most

$$\left\lceil \frac{1}{2} \cdot \left(\frac{n}{|Q_i|} + 2\right)\right\rceil \leq \frac{n}{2 \cdot |Q_i|} + \frac{2}{2} + 1 \leq \frac{n}{|Q_{i+1}|} + 2. \quad \square$$

**Bounding the Running Time.** We now bound the running time of our algorithm. The running time of Phase I of our algorithm is $O(n\ell) = \tilde{O}(n)$, since it takes $O(n)$ time to construct each partition $Q_i$ given the partition $Q_{i-1}$. The running time of Phase III of our algorithm is $\tilde{O}(nk)$, since constructing the mapping $w_0$ takes $O(nk)$ time and running the Mettu-Plaxton algorithm on an input of size $2k$ takes $\tilde{O}(k^2)$ time. Thus, we now focus on bounding the running time of Phase II.

We can first observe that the running time of the $i^{th}$ iteration in Phase II is dominated by the total time taken to handle the calls to the algorithm RES-GREEDY. In the first iteration (when $i = \ell$), we make $|Q_\ell|$ many calls to RES-GREEDY, each one on a subspace of size at most $n/|Q_\ell| + 2$ (by Lemma 5.6). Thus, by Theorem 4.1, the time taken to handle these calls is at most

$$\tilde{O}(1) \cdot \left(\frac{n}{|Q_\ell|} + 2\right)^2 \cdot |Q_\ell| \leq \tilde{O}(1) \cdot \left(\frac{n}{|Q_\ell|}\right)^2 \cdot |Q_\ell| \leq \tilde{O}(1) \cdot \frac{n^2}{|Q_\ell|} \leq \tilde{O}(1) \cdot \frac{n^2}{2^\ell} \leq \tilde{O}(nk),$$

---

[9]Note that we do not necessarily guarantee that all of the sets in these partitions are non-empty.

where the first inequality follows from the fact that $n/|Q_\ell| \geq 1$, the third from Lemma 5.6, and the fourth since $2^\ell \geq n/k$. It follows that the time taken to handle these calls to RES-GREEDY is $\tilde{O}(nk)$. For each subsequent iteration (when $0 \leq i < \ell$), we make $|Q_i|$ many calls to RES-GREEDY, each one on a subspace $(X, w, d)$ of size at most $n/|Q_i| + 2$ (by Lemma 5.6), where the solution is restricted to the set $X \cap V_{i+1}$, which has size at most $|X \cap V_{i+1}| = |S_{X_1} \cup S_{X_2}| \leq 4k$, where $\mathcal{P}(X) = \{X_1, X_2\}$ and $S_{X_1}$ and $S_{X_2}$ are computed in the previous iteration. It follows from Theorem 4.1 that the time taken to handle these calls is at most $\tilde{O}(1) \cdot (n/|Q_i| + 2) \cdot 4k \cdot |Q_i| \leq \tilde{O}(nk)$. It follows that the total time taken to handle these calls to RES-GREEDY during the $i^{th}$ iteration of Phase II is $\tilde{O}(nk)$. Hence, the total time spent handling calls to RES-GREEDY is $\ell \cdot \tilde{O}(nk) = \tilde{O}(nk)$. The running time of our algorithm follows.

# References

[ACS22]    Sepehr Assadi, Andrew Chen, and Glenn Sun. Deterministic graph coloring in the streaming model. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 261–274. ACM, 2022.

[AGK+04]   Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[AJM09]    Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. *Advances in neural information processing systems*, 22, 2009.

[ANFSW19]  Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, 49(4):FOCS17–97, 2019.

[BCF25]    Sayan Bhattacharya, Martín Costa, and Ermiya Farokhnejad. Fully dynamic $k$-median with near-optimal update time and recourse. In *57th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2025. (To Appear).

[BCG+24]   Sayan Bhattacharya, Martín Costa, Naveen Garg, Silvio Lattanzi, and Nikos Parotsidis. Fully dynamic $k$-clustering with fast update time and small recourse. In *65th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2024.

[BCLP23]   Sayan Bhattacharya, Martín Costa, Silvio Lattanzi, and Nikos Parotsidis. Fully dynamic $k$-clustering in $\tilde{O}(k)$ update time. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[BEF+23]   MohammadHossein Bateni, Hossein Esfandiari, Hendrik Fichtenberger, Monika Henzinger, Rajesh Jayaram, Vahab Mirrokni, and Andreas Wiese. Optimal fully dynamic $k$-center clustering for adaptive and oblivious adversaries. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2677–2727. SIAM, 2023.

[BPR+17]   Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Transactions on Algorithms (TALG)*, 13(2):1–31, 2017.

[CGTS99]     Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10, 1999.

[Cha16]      Ching-Lueh Chang. Metric 1-median selection: Query complexity vs. approximation ratio. In *Computing and Combinatorics - 22nd International Conference, COCOON 2016, Ho Chi Minh City, Vietnam, August 2-4, 2016, Proceedings*, volume 9797 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 2016.

[CHP+19]     Vincent Cohen-Addad, Niklas Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. Fully dynamic consistent facility location. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3250–3260, 2019.

[CKY06]      Marek Chrobak, Claire Kenyon, and Neal E. Young. The reverse greedy algorithm for the metric *k*-median problem. *Inf. Process. Lett.*, 97(2):68–72, 2006.

[COP03]      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.

[CSS23]      Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. Deterministic clustering in high dimensional spaces: Sketches and approximation. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1105–1130. IEEE, 2023.

[DHS24]      Max Dupré la Tour, Monika Henzinger, and David Saulpic. Fully dynamic k-means coreset in near-optimal update time. In *32nd Annual European Symposium on Algorithms, ESA 2024*, volume 308 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[DS24]       Max Dupré la Tour and David Saulpic. Almost-linear time approximation algorithm to euclidean k-median and k-means. *CoRR*, abs/2407.11217, 2024.

[GMMO00]     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.

[Gon85]      Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[GT08]       Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[HK20]       M. Henzinger and S. Kale. Fully-dynamic coresets. In *ESA*, 2020.

[HLRW24]     Monika Henzinger, Jason Li, Satish Rao, and Di Wang. Deterministic near-linear time minimum cut in weighted graphs. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3089–3139. SIAM, 2024.

[HLS24]     Bernhard Haeupler, Yaowei Long, and Thatchaphol Saranurak. Dynamic deterministic constant-approximate distance oracles with n$^\epsilon$ worst-case update time. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 2033–2044. IEEE, 2024.

[HN79]      Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discret. Appl. Math.*, 1(3):209–215, 1979.

[JV01]      Kamal Jain and Vijay V Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM (JACM)*, 48(2):274–296, 2001.

[MN20]      Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 496–509. ACM, 2020.

[MP00]      Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 339–348, 2000.

[MP02]      Ramgopal R. Mettu and C. Greg Plaxton. Optimal time bounds for approximate clustering. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 344–351, 2002.

[NS16]      Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Trans. Algorithms*, 12(1):7:1–7:15, 2016.

# A    The Algorithm of [GMMO00] (Proof of Theorem 3.2)

In this section, we prove Theorem 3.2, which we restate below.

**Theorem A.1.** *There is a deterministic algorithm for $k$-median that, given a metric space of size $n$, computes a $\mathrm{poly}(\log(n/k)/\log\delta)$-approximate solution in $\tilde{O}(nk\delta)$ time, for any $2 \le \delta \le n/k$.*

## A.1    Preliminaries

In this section, for ease of notation, we consider solutions to the $k$-median problem to be mappings instead of subsets of points. More precisely, we denote a solution to the $k$-median problem on $(V, w, d)$ by a mapping $\sigma : V \longrightarrow S$, where $S = \sigma(V)$ is the set of centres of size at most $k$, and each $x \in V$ is assigned to center $\sigma(x)$. We denote the cost of this solution by $\texttt{cost}(\sigma, V, w) := \sum_{x\in V} w(x) d(x, \sigma(x))$.

**The Mettu-Plaxton Algorithm.** This algorithm uses the $\tilde{O}(n^2)$ time algorithm of [MP00] as a black box, which we refer to as MP-ALG for convenience.[10] The following theorem summarizes the properties of MP-ALG.

**Theorem A.2** ([MP00]). *There exists a deterministic algorithm MP-ALG that, given a metric space of size $n$, returns a $O(1)$-approximation to the $k$-median problem in at most $\tilde{O}(n^2)$ time.*

For notational convenience, we denote the approximation ratio and the hidden polylogarithmic function in the running time of MP-ALG by $\alpha$ and $A$ respectively. Thus, given a metric space of size $n$, MP-ALG returns an $\alpha$-approximation in time at most $A \cdot n^2$.

## A.2    The Algorithm

Let $(V, w, d)$ be a metric space of size $n$, $k \le n$ be an integer and $\delta > 1$ be a parameter. We also define values $\ell := \lceil \log_2(\log(n/k)/\log\delta) \rceil$, $\gamma := n/k$, and $q_i := \lceil \gamma^{1/2^i} \rceil$ for each $i \in [\ell]$, which we use to describe the algorithm. The algorithm works in 2 *phases*, which we describe below.

**Phase I:** In the first phase of the algorithm, we construct a sequence of partitions $Q_0, \ldots, Q_\ell$ of the metric space $V$, such that the partition $Q_i$ is a *refinement* of the partition $Q_{i-1}$.[11] We start off by setting $Q_0 := \{V\}$. Subsequently, for each $i = 1, \ldots, \ell$, we construct the partition $Q_i$ as follows:

> Initialize $Q_i \leftarrow \varnothing$. Then, for each $X \in Q_{i-1}$, arbitrarily partition $X$ into subsets $X_1, \ldots, X_{q_i}$ such that $\big| |X_j| - |X_{j'}| \big| \le 1$ for each $j, j' \in [q_i]$, and add these subsets to $Q_i$.

**Phase II:** The second phase of the algorithm proceeds in *iterations*, where we use the partitions $\{Q_i\}_i$ to compute the solution in a bottom-up manner. Let $V_{\ell+1}$ and $w_{\ell+1}$ denote the set of points $V$ and the weight function $w$ respectively. For each $i = \ell, \ldots, 0$, the algorithm constructs $V_i$ as follows:

> For each $X \in Q_i$, let $\sigma_X$ be the solution obtained by running MP-ALG on the metric space $(X \cap V_{i+1}, w_{i+1}, d)$ and $S_X := \sigma_X(X \cap V_{i+1})$. For each center $y \in S_X$, let $w_i(y) := \sum_{x \in \sigma_X^{-1}(y)} w_{i+1}(x)$ be the total weight (w.r.t. $w_{i+1}$) of the points assigned to $y$ in $X$ by $\sigma_X$. Let $V_i := \bigcup_{X \in Q_i} S_X$.

---

[10]We can also use any other $O(1)$-approximate algorithm that runs in time $\tilde{O}(n^2)$.
[11]i.e. for each element $X \in Q_{i-1}$, there are elements $X_1, \ldots, X_q \in Q_i$ such that $X = X_1 \cup \cdots \cup X_q$.

**Output:** For each $i \in [0, \ell]$, let $\sigma_i : V_{i+1} \longrightarrow V_i$ denote the mapping obtained by taking the union of the mappings $\{\sigma_X\}_{X \in Q_i}$.[12] The output of the algorithm is the mapping $\sigma : V \longrightarrow V_0$, which we define as the composition $\sigma := \sigma_0 \circ \cdots \circ \sigma_\ell$ of the $\sigma_i$.

## A.3 Analysis

We now analyze the algorithm by bounding its approximation ratio and running time. We begin by proving the following lemmas, which summarize the relevant properties of the partitions constructed in Phase I of the algorithm.

**Lemma A.3.** *For each $i \in [\ell]$, the set $Q_i$ is a partition of $V$ into $\prod_{j=1}^{i} q_j$ many subsets of size at most $n/|Q_i| + i$. Furthermore, $Q_i$ is a refinement of $Q_{i-1}$.*

*Proof.* We define $Q_0$ as $\{V\}$, which is a trivial partition of $V$. Now, suppose that this statement holds for the partition $Q_i$, where $0 \le i < \ell$. The algorithm constructs $Q_{i+1}$ by taking each $X \in Q_i$ and further partitioning $X$ into subsets $X_1, \ldots, X_{q_{i+1}}$, such that difference in the sizes of any two of these subsets is at most 1. Clearly, the partition $Q_{i+1}$ is a refinement of the partition $Q_i$. We can also observe that the number of subsets in the partition $Q_{i+1}$ is $q_{i+1} \cdot |Q_i| = q_{i+1} \cdot \prod_{j=1}^{i} q_j = \prod_{j=1}^{i+1} q_j$.[13] Finally, since each subset $X \in Q_i$ has size at most $n/|Q_i| + i$, it follows that each subset in $Q_{i+1}$ has size at most

$$\left\lceil \frac{1}{q_{i+1}} \cdot \left( \frac{n}{|Q_i|} + i \right) \right\rceil \le \frac{n}{q_{i+1} \cdot |Q_i|} + \frac{i}{q_{i+1}} + 1 \le \frac{n}{|Q_{i+1}|} + (i+1). \qquad \square$$

**Claim A.4.** *For each $i \in [\ell]$, we have that $\gamma^{1-1/2^i} \le \prod_{j=1}^{i} q_j \le e^i \cdot \gamma^{1-1/2^i}$.*

*Proof.* Let $i \in [\ell]$. For the lower bound, we can see that

$$\prod_{j=1}^{i} q_j = \prod_{j=1}^{i} \lceil \gamma^{1/2^j} \rceil \ge \prod_{j=1}^{i} \gamma^{1/2^j} = \gamma^{\sum_{j=1}^{i} 1/2^j} = \gamma^{1-1/2^i}. \tag{8}$$

For the upper bound, we use the fact that $\lceil x \rceil \le x + 1 = x(1 + 1/x)$ to get that

$$\prod_{j=1}^{i} q_j = \prod_{j=1}^{i} \lceil \gamma^{1/2^j} \rceil \le \prod_{j=1}^{i} \gamma^{1/2^j} \cdot \left( 1 + \gamma^{-1/2^j} \right) = \left( \prod_{j=1}^{i} \gamma^{1/2^j} \right) \cdot \left( \prod_{j=1}^{i} \left( 1 + \gamma^{-1/2^j} \right) \right). \tag{9}$$

It follows from Equation (8) that $\prod_{j=1}^{i} \gamma^{1/2^j} = \gamma^{1-1/2^i}$. We can also see that

$$\prod_{j=1}^{i} \left( 1 + \gamma^{-1/2^j} \right) \le \prod_{j=1}^{i} \exp\left( \gamma^{-1/2^j} \right) = \exp\left( \sum_{j=1}^{i} \gamma^{-1/2^j} \right) \le e^i.$$

Thus, combining these upper bounds with Equation (9), it follows that $\prod_{j=1}^{i} q_j \le e^i \cdot \gamma^{1-1/2^i}$. $\quad\square$

---

[12]Note that, since the domains of the mappings $\{\sigma_X\}_{X \in Q_i}$ partition $V_{i+1}$, their union is well defined.

[13]Note that we do not necessarily guarantee that all of the sets in these partitions are non-empty.

## Approximation Ratio

To analyze the approximation ratio of the algorithm, we use the following lemma of [GMMO00], which shows that we can use a good bicriteria approximation for $k$-median as a 'sparsifier' for the underlying metric space. A proof of this lemma using the same notation as our paper can be found in [BCG$^+$24].

**Lemma A.5** (Lemma 10.3, [BCG$^+$24]). *Let $(V, w, d)$ be a metric space, $\sigma : V \longrightarrow V'$ be a mapping such that $\mathsf{cost}(\sigma, V, w) \leq \beta \cdot \mathrm{OPT}_k(V, w)$, and define $w'(y) := \sum_{x \in \sigma^{-1}(y)} w(x)$ for all $y \in V'$. Given a mapping $\pi : V' \longrightarrow S$ such that $\mathsf{cost}(\pi, V', w') \leq \alpha \cdot \mathrm{OPT}_k(V', w')$, we have that*

$$\mathsf{cost}(\pi \circ \sigma, V, w) \leq (2\alpha + (1 + 2\alpha)\beta) \cdot \mathrm{OPT}_k(V, w).$$

For each $i \in [0, \ell]$, let $\sigma_i'$ denote the mapping $\sigma_i \circ \cdots \circ \sigma_\ell$. Note that the output of the algorithm is precisely $\sigma_0'$. We use Lemma A.5 to inductively bound the approximation ratio of each $\sigma_i'$. In particular, we prove the following lemma.

**Lemma A.6.** *For each $0 \leq i \leq \ell$, $\mathsf{cost}(\sigma_i', V, w) \leq (9\alpha)^{\ell+1-i} \cdot \mathrm{OPT}_k(V, w)$.*

*Proof.* Let $\sigma_{\ell+1}'$ denote the identity mapping on $V$. Then, we clearly have that $\mathsf{cost}(\sigma_{\ell+1}', V, w) = 0$. Now, let $i \in [0, \ell]$ and suppose that

$$\mathsf{cost}(\sigma_{i+1}', V, w) \leq (9\alpha)^{\ell-i} \cdot \mathrm{OPT}_k(V, w). \tag{10}$$

Let $\sigma_{i+1}^\star$ denote an optimal solution to the $k$-median problem in the metric space $(V_{i+1}, w_{i+1}, d)$. We can upper bound the cost of $\sigma_i$ (in the space $(V_{i+1}, w_{i+1}, d)$) by

$$\begin{aligned}
\mathsf{cost}(\sigma_i, V_{i+1}, w_{i+1}) &= \sum_{X \in Q_i} \mathsf{cost}(\sigma_X, X \cap V_{i+1}, w_{i+1}) \\
&\leq \sum_{X \in Q_i} 2\alpha \cdot \mathsf{cost}(\sigma_{i+1}^\star, X \cap V_{i+1}, w_{i+1}) \\
&= 2\alpha \cdot \mathsf{cost}(\sigma_{i+1}^\star, V_{i+1}, w_{i+1}) \\
&= 2\alpha \cdot \mathrm{OPT}_k(V_{i+1}, w_{i+1}), \tag{11}
\end{aligned}$$

where the first and third lines follow from the fact that $\{X \cap V_{i+1} \mid X \in Q_i\}$ partitions $V_{i+1}$ and the second line from $\sigma_X$ being an $\alpha$-approximate solution in the metric space $(X \cap V_{i+1}, w_{i+1}, d)$. We note that the extra factor of 2 in the third line follows from the fact that $\sigma_{i+1}^\star(X \cap V_{i+1})$ might contain points that are not in $X \cap V_{i+1}$.

Now, since we have that $\sigma_i' = \sigma_i \circ \sigma_{i+1}'$, we can apply Lemma A.5 using the upper bounds on $\mathsf{cost}(\sigma_{i+1}', V, w)$ and $\mathsf{cost}(\sigma_i, V_{i+1}, w_{i+1})$ given in Equations (10) and (11) to get that

$$\mathsf{cost}(\sigma_i', V, w) \leq (4\alpha + (1 + 4\alpha) \cdot (9\alpha)^{\ell-i}) \cdot \mathrm{OPT}_k(V, w) \leq (9\alpha)^{\ell+1-i} \cdot \mathrm{OPT}_k(V, w). \qquad \square$$

It follows from Lemma A.6 by setting $i = 0$ that

$$\mathsf{cost}(V_0, V, w) = 2^{O(\ell)} \cdot \mathrm{OPT}_k(V, w) = \mathrm{poly}(\log(n/k)/\log \delta) \cdot \mathrm{OPT}_k(V, w).$$

22

**Running Time**

The running time of Phase I of the algorithm is $O(n\ell) = \tilde{O}(n)$, since it takes $O(n)$ time to construct each partition $Q_i$ given the partition $Q_{i-1}$. Thus, we now focus on bounding the running time of Phase II.

We can first observe that the running time of the $i^{th}$ iteration in Phase II is dominated by the total time taken to handle the calls to the algorithm MP-ALG. In the first iteration (when $i = \ell$), we make $|Q_\ell|$ many calls to MP-ALG, each one on a subspace of size at most $n/|Q_\ell| + \ell$ (by Lemma A.3). Thus, by Theorem A.2, the time taken to handle these calls is at most

$$A \cdot \left(\frac{n}{|Q_\ell|} + \ell\right)^2 \cdot |Q_\ell| \leq A\ell^2 \cdot \left(\frac{n}{|Q_\ell|}\right)^2 \cdot |Q_\ell| = A\ell^2 \cdot \frac{n^2}{|Q_\ell|} = A\ell^2 \cdot \frac{n^2}{\prod_{j=1}^{\ell} q_j} \leq A\ell^2 \cdot \frac{n^2}{\gamma^{1-1/2^\ell}}, \quad (12)$$

where the first inequality follows from the fact that $n/|Q_\ell| \geq 1$ and $\ell \geq 1$, the last equality follows from Lemma A.3, and the last inequality follows from Claim A.4. We can now upper bound the RHS of Equation (12) by

$$A\ell^2 \cdot \frac{n^2}{\gamma^{1-1/2^\ell}} = A\ell^2 \cdot n^2 \cdot \frac{k}{n} \cdot \gamma^{1/2^\ell} = A\ell^2 \cdot nk \cdot \gamma^{2^{-\ell}} \leq A\ell^2 \cdot nk \cdot \delta. \quad (13)$$

Thus, the time taken to handle these calls to MP-ALG is $\tilde{O}(nk\delta)$. For each subsequent iteration (when $0 \leq i < \ell$), we make $|Q_i|$ many calls to MP-ALG, each one on a subspace $(X \cap V_{i+1}, w_{i+1}, d)$ of size at most $q_{i+1}k$ since $|X \cap V_{i+1}| = |\bigcup_{j=1}^{q_{i+1}} S_{X_j}| \leq q_{i+1}k$, where $X_1, \ldots, X_{q_{i+1}}$ are the subsets that $X$ is partitioned into, and each $S_{X_j}$ is the solution computed on the subspace $(X_j \cap V_{i+2}, w_{i+2}, d)$ in the previous iteration. It follows that the time taken to handle these calls is at most

$$A \cdot (q_{i+1}k)^2 \cdot |Q_i| = A \cdot (q_{i+1}k)^2 \cdot \prod_{j=1}^{i} q_j = A \cdot k^2 \cdot q_{i+1} \cdot \prod_{j=1}^{i+1} q_j$$

$$\leq A \cdot k^2 \cdot 2\gamma^{1/2^{i+1}} \cdot e^i \gamma^{1-1/2^{i+1}} = 2Ae^i \cdot k^2 \cdot \gamma = 2Ae^i \cdot nk.$$

where the first equality follows from Lemma A.3 and the first inequality follows from Claim A.4 and the fact that $q_{i+1} \leq 2\gamma^{1/2^{i+1}}$. Since $\ell = O(\log\log n)$, we get that $e^i \leq e^\ell = \tilde{O}(1)$ and it follows that the total time taken to handle these calls to MP-ALG is $\tilde{O}(nk)$. Consequently, the running time of the algorithm is $\tilde{O}(nk\delta) + \ell \cdot \tilde{O}(nk) = \tilde{O}(nk\delta)$.

## A.4 Extension to $k$-Means

It is straightforward to extend this algorithm to the $k$-means problem, where the clustering objective is $\sum_{x \in V} w(x) \cdot d(x, S)^2$ instead of $\sum_{x \in V} w(x) \cdot d(x, S)$. In particular, we get the following theorem.

**Theorem A.7.** *There is a deterministic algorithm for $k$-means that, given a metric space of size $n$, computes a $\text{poly}(\log(n/k)/\log\delta)$-approximate solution in $\tilde{O}(nk\delta)$ time, for any $2 \leq \delta \leq n/k$.*

We define the *normalized $k$-means* objective as $\text{cost}_2(S) := \left(\sum_{x \in V} w(x) \cdot d(x, S)^2\right)^{1/2}$. As pointed out by [BCG+24], for technical reasons, it is easier to work with this notion of normalized cost. By observing that a solution $S$ is an $\alpha^2$-approximation to $k$-means if and only if it is an $\alpha$-approximation to normalized $k$-means, we can assume w.l.o.g. that we are working with the normalized objective function.

Since the Mettu-Plaxton algorithm [MP02] can also be used to give a $O(1)$-approximation to the normalized $k$-means problem, this algorithm works for normalized $k$-means without any modification. Thus, the running time guarantees extend immediately. Furthermore, [BCG+24] show that the exact statement of Lemma A.5 holds for the normalized $k$-means objective, i.e. replacing cost with $\texttt{cost}_2$. Using this lemma, it is easy to see that the analysis of the approximation also extends with no modifications.

# B  Our Lower Bound for Deterministic $k$-Median

In this section, we prove the following theorem.

**Theorem B.1.** *For every $\delta \geq 1$, any deterministic algorithm for the $k$-median problem that has a running time of $O(kn\delta)$ on a metric space of size $n$ has an approximation ratio of*

$$\Omega\left(\frac{\log n}{\log \log n + \log k + \log \delta}\right).$$

Theorem 1.2 follows from Theorem B.1 by setting $\delta = \tilde{O}(1)$.

## B.1  The Proof Strategy

Our proof of Theorem B.1 is a modification and slight simplification of a lower bound given in the work of [BEF+23], which provides lower bounds for various $k$-clustering problems in different computational models.

Our proof uses the following approach: Consider any deterministic algorithm ALG for the $k$-median problem. Given a metric space $(V, d)$ as input, this algorithm can only access information about the metric space by querying the distance $d(x, y)$ between two points $x$ and $y$ in $V$. We design an adversary $\mathcal{A}$ which takes as input a deterministic algorithm ALG and constructs a metric space $(V, \mathfrak{d})$ on which the algorithm ALG has a bad approximation ratio. The adversary does this by running the algorithm ALG on a set of points $V$ and *adaptively* answering the distance queries made by the algorithm in a specific way, where the queries made by the algorithm and the responses given by the adversary are a function of the previous queries and responses. Throughout this process, the adversary constructs a metric $\mathfrak{d}$ on the point set $V$ which is consistent with the responses that it has given to the distance queries and also guarantees that the solution $S$ output by ALG at the end of this process has a bad approximation ratio compared to the optimal solution in $(V, \mathfrak{d})$. Since the algorithm ALG is deterministic, its output when run on the metric space $(V, \mathfrak{d})$ is the same as the solution $S$ that it outputs during this process.

## B.2  The Adversary $\mathcal{A}$

The adversary $\mathcal{A}$ begins by creating a set of $n$ points $V$, which it feeds to an instance of ALG as its input.[14] Whenever the algorithm ALG attempts to query the distance between two points, the adversary determines the response to the query using the strategy that we describe below. We begin by describing the notation that we use throughout the rest of this section.

**Notation.** Throughout this section, we use parameters $\delta > 1$ and $M := 10k\delta \log n$. The parameter $\delta$ is chosen such that the query complexity of the deterministic algorithm is at most $nk\delta$. Given a

---

[14]We remark that the algorithm ALG is *not* being given a metric space as input, since there is no metric associated with the points in $V$ at this point.

weighted graph $H$ and two nodes $u$ and $v$ of $H$, we denote the weight of the edge $(u, v)$ by $w(u, v)$ and denote the weight of the shortest path between $u$ and $v$ in $H$ by $\texttt{dist}_H(u, v)$.

**The Graph $G$.** The adversary $\mathcal{A}$ maintains a simple, undirected graph $G$ which it uses to keep track of the queries that have already been made. The graph $G$ has $n + 1$ nodes: one special node $g^\star$ and $n$ nodes $v_x$ corresponding to each $x \in V$. At any point in time, each node in $G$ has a *status* which is either *open* or *closed*. All of the nodes are initially open except $g^\star$. Initially, the graph $G$ consists of $n$ edges of weight $\log_M n$ between $g^\star$ and each of the other nodes $v_x$ in $G$. We note that the point $g^\star$ ensures that the distance between any two nodes in $G$ is always at most $2\log_M n$.

**The Auxiliary Graph $\widehat{G}$.** At any point in time, we denote by $\widehat{G}$ the graph derived from $G$ by adding edges of weight 1 between each pair of open nodes in $G$. For instance, the graph $\widehat{G}$ initially consists of a clique of size $n$ made out of the nodes $\{v_x\}_{x \in V}$, all of whose edges have weight 1, together with the node $g^\star$ and edges of weight $\log_M n$ between $g^\star$ and the nodes $\{v_x\}_{x \in V}$ in the clique.

### Handling a Query

We now describe how the adversary $\mathcal{A}$ handles a query $\langle x, y \rangle$ and updates the graph $G$. Depending on the status of nodes $v_x$ and $v_y$, $\mathcal{A}$ does one of the following.

**Case 1.** If there already exists an edge between the nodes $v_x$ and $v_y$ in $G$ (which means the distance between $x$ and $y$ is already fixed), the adversary returns the weight $w(v_x, v_y)$ as the distance between $x$ and $y$.

**Case 2.** If both of $v_x$ and $v_y$ are open, $\mathcal{A}$ reports the distance between $x$ and $y$ as 1. It then adds an edge in $G$ between $v_x$ and $v_y$ with weight $w(v_x, v_y) = 1$. Finally, if there are any open nodes of degree at least $M$, $\mathcal{A}$ sets the status of these nodes to closed.

**Case 3.** If at least one of $v_x$ or $v_y$ is closed, the adversary considers the auxiliary graph $\widehat{G}$ (corresponding to the current graph $G$). $\mathcal{A}$ then reports the distance of $x$ and $y$ as the weighted shortest path between $v_x$ and $v_y$ in $\widehat{G}$, i.e. as $\texttt{dist}_{\widehat{G}}(v_x, v_y)$. This shortest path contains at most one edge between two open nodes (otherwise, there would be a shortcut since the subgraph of $\widehat{G}$ on open nodes is a clique with all edges having weight 1). If such an edge $(u, v)$ between two open nodes within this shortest path exists, $\mathcal{A}$ adds an edge between $u$ and $v$ in $G$ of weight $w(u, v) = 1$. Then, $\mathcal{A}$ adds an edge between $v_x$ and $v_y$ in $G$ of weight $\texttt{dist}_{\widehat{G}}(v_x, v_y)$ (the reported distance between $x$ and $y$). Finally, if there are any open nodes of degree at least $M$, $\mathcal{A}$ sets the status of these nodes to closed.

**Constructing the Final Graph and Metric.** After at most $nk\delta$ many queries, the deterministic algorithm returns a subset $S \subseteq V$ of size $k$ as its output.[15] Once this happens, the adversary proceeds to make some final modifications to the graph $G$. Namely, $\mathcal{A}$ pretends that the distance of each point in $S$ to every other point in $V$ is queried. In other words, $\mathcal{A}$ makes the same changes to $G$ that would occur if $\textsc{Alg}$ had queried $\langle x, y \rangle$ for each $y \in S$ and each $x \in V$. The order of these artificial queries is arbitrary. We denote this final graph by $G_f$.

Finally, we define the metric $\mathfrak{d}$ on $V$ to be the weighted shortest path metric in $\widehat{G}_f$, i.e. we define $\mathfrak{d}(x, y) := \texttt{dist}_{\widehat{G}_f}(v_x, v_y)$ for each $x, y \in V$. The adversary then returns the metric space $(V, \mathfrak{d})$, which is an instance on which $\textsc{Alg}$ returns a solution with a bad approximation ratio.

---

[15]We can assume w.l.o.g. that the set $S$ contains exactly $k$ points, since we can add extra arbitrarily if $|S| \leq k$.

## B.3 Analysis

We show that the final metric $(V, \mathfrak{d})$ is consistent with the answers given by the adversary to the queries made by the deterministic algorithm. In other words, if we run ALG on this metric, it will return the same solution $S$. We defer the proof of the following lemma to Appendix B.4.

**Lemma B.2** (Consistency of Metric). *For each $x$ and $y$ where $\langle x, y \rangle$ is **queried**, the distance of points $x$ and $y$ in the final metric (i.e. $\mathtt{dist}_{\widehat{G}_f}(v_x, v_y)$) equals the value returned by $\mathcal{A}$ in response to the query $\langle x, y \rangle$ (i.e. $w(v_x, v_y)$ in $\widehat{G}_f$).*

We proceed with the analysis of the approximation ratio. We show that the cost of $S$ as a $k$-median solution in this space is comparably higher than the cost of the optimum $k$-median solution in this space. In particular, we show that the cost of any arbitrary set of $k$ centers containing at least one point corresponding to an **open** node is small.

**Claim B.3.** *For each $z \in S$ and $1 \leq i \leq \log_M(n)$, there are at most $M \cdot (M-1)^{i-1}$ points whose distance to $z$ is equal to $i$.*

We defer the proof of this claim to Appendix B.5.

**Lemma B.4.** *The cost of $S$ as a $k$-median solution is at least $(n/2) \cdot \lfloor \log_M n \rfloor$.*

*Proof.* Assume $r$ is the biggest integer such that $M^r \leq n$, i.e. $r = \lfloor \log_M n \rfloor$. According to Claim B.3, for each $1 \leq i \leq r-1$, there are at most $M(M-1)^{i-1}$ many points with distance $i$ to $z$ for any arbitrary $z \in S$. As a result, the total number of points whose distance to $z$ is at most $r-1$ is bounded by

$$1 + M + M(M-1) + M(M-1)^2 + \cdots + M(M-1)^{r-2} \leq 2M^{r-1}.$$

Since $|S| = k$, there are at most $2kM^{r-1}$ points whose distance to $S$ is less than or equal to $r$. We conclude there are at least $n - 2kM^{r-1} \geq n - 2kn/M = (1 - 2k/M)n$ points whose distance to $S$ is greater than or equal to $r$. Hence, the cost of $S$ is at least $(1 - 2k/M)n \cdot r$. Note that $(1 - 2k/M) = (1 - 1/(5\delta \log n)) \geq 1/2$, which implies $(1 - 2k/M)n \cdot r \geq (n/2) \cdot \lfloor \log_M n \rfloor$. $\square$

**Claim B.5.** *The number of closed nodes in $G_f$ is at most $(10k\delta/M)n$.*

*Proof.* According to the building procedure of $G$, every time that $\mathcal{A}$ answers a query, at most 2 edges are added to $G$. This means that the total number of edges in $G_f$ is at most $2nk\delta + 2nk + n$. The $2nk\delta$ term is for the normal queries of the algorithm. The $2nk$ term is because, after the termination of the algorithm, the adversary queries at most $kn$ distances between $S$ and all other points, which adds at most $2nk$ edges to the graph in total. The $n$ term is because the initial graph $G$ consists of $n$ edges. Now, assume that we have $C$ many closed nodes in $G_f$. Since the degree of each closed node is at least $M$, we have

$$MC/2 \leq \text{ total number of edges in } G_f \leq 2nk\delta + 2nk + n \leq 5nk\delta.$$

As a result, $C \leq (10k\delta/M)n$. $\square$

**Lemma B.6.** *The cost of any arbitrary set of $k$ centers containing at least 1 open node (in the final graph $G_f$) is at most $3n$.*

*Proof.* Let $\alpha := 10k\delta/M = 1/\log n < 1$. According to Claim B.5, there are at most $\alpha n$ many closed nodes in $G$. So, at least one open node exists. Let $S^\star$ be any set of $k$ centers containing a point $x^\star$ such that $v_{x^\star}$ is an open node in $G_f$. The distance of any closed node to $v_{x^\star}$ is at most $2\log_M n$, since there is always the path of weight at most $2\log_M n$ between $v_{x^\star}$ and any closed node passing by $g^\star$. The distance between any open node and $v_{x^\star}$ is at most 1 according to the definition of the final metric. Hence, the total cost of $S^\star$ which is at most the cost of assigning all of the points only to $x^\star$ is at most

$$(\alpha n) \cdot 2\log_M n + ((1-\alpha)n) \cdot 1 = 2n \cdot \frac{\log_M n}{\log n} + (1-\alpha) \cdot n \le 3n. \qquad \square$$

Now, we are ready to prove Theorem B.1. The approximation ratio of the algorithm according to Lemma B.4 and Lemma B.6 is at least

$$\frac{(n/2)\lfloor \log_M n \rfloor}{3n} = \Omega(\log_M n).$$

Here, we assumed that $M \le n$. Otherwise, $\lfloor \log_M n \rfloor = 0$. Note that in the case where $M > n$, the final lower bound in Theorem B.1 becomes a constant and the theorem is obvious in this case since the approximation ratio of every algorithm for $k$-median is at least 1. With some more calculations, we conclude

$$\log_M n = \frac{\log n}{\log(10k\delta \log n)} = \Omega\left(\frac{\log n}{\log\log n + \log k + \log \delta}\right).$$

## B.4   Proof of Lemma B.2 (Consistency of The Metric)

By induction on the number of queries, we show that, **at any point in time**, if there is an edge between nodes $v_x$ and $v_y$ in $G$, then $w(v_x, v_y) = \mathtt{dist}_{\widehat{G}}(v_x, v_y)$. Assume $G_1$ is the current graph after some queries (possibly zero, at the very beginning) and that it satisfies this condition. Let $\langle x, y \rangle$ be the new query. We have the following three cases.

**Case 1.** There is already an edge between $v_x$ and $v_y$. According to the strategy of $\mathcal{A}$ in this case, $G_1$ is not going to change and still satisfies the property required by the lemma.

**Case 2.** If both $v_x$ and $v_y$ are open, then an edge of weight 1 is added to $G_1$. Let $G_2$ be the new graph. Since, $v_x$ and $v_y$ are open in $G_1$, according to the definition of $\widehat{G}_1$, the edge of weight 1 between $v_x$ and $v_y$ was already in $\widehat{G}_1$. So, the edges of $\widehat{G}_2$ are a subset of the edges of $\widehat{G}_1$. Note that after the addition of $(v_x, v_y)$, the degree of $v_x$ or $v_y$ might become greater than or equal to $M$ and the adversary will mark them as closed in $G_2$. So, it is possible that $\widehat{G}_2$ has less edges than $\widehat{G}_1$ but not more edges, which concludes for each pair of arbitrary nodes $v_p$ and $v_q$, $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q) \ge \mathtt{dist}_{\widehat{G}_1}(v_p, v_q)$. In particular, for each pair $v_p, v_q$, such that $\langle p, q \rangle$ has been queried, we have that

$$\mathtt{dist}_{\widehat{G}_2}(v_p, v_q) \ge \mathtt{dist}_{\widehat{G}_1}(v_p, v_q). \tag{14}$$

Now, according to the induction hypothesis, $\mathtt{dist}_{\widehat{G}_1}(v_p, v_q) = w(v_p, v_q)$ and this edge is present in $\widehat{G}_2$ which can be considered as a feasible path between $v_p$ and $v_q$ in $\widehat{G}_2$. Hence, $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q) \le w(v_p, v_q) = \mathtt{dist}_{\widehat{G}_1}(v_p, v_q)$. Together with Equation (14), $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q) = \mathtt{dist}_{\widehat{G}_1}(v_p, v_q) = w(v_p, v_q)$ in $\widehat{G}_2$ as well.

**Case 3.** At least one of $v_x$ and $v_y$ is closed. In this case, the adversary sets $w(v_x, v_y) = \mathtt{dist}_{\widehat{G}_1}(v_x, v_y)$. There might also be a new edge of weight 1 added to $G_1$ between two open nodes. Let $G_2$ be the new graph. We have to show that for each pair of nodes $v_p$ and $v_q$ such that $\langle p, q \rangle$

has been queried, we have $w(v_p, v_q) = \mathtt{dist}_{\widehat{G}_2}(v_p, v_q)$. With a similar argument as the previous case, we can see that the only edge that $\widehat{G}_2$ might contain but $\widehat{G}_1$ does not contain is the new edge $(v_x, v_y)$ (In the case where the adversary also adds an edge of weight 1 between two open nodes of $G_1$, we know that this edge is already present in $\widehat{G}_1$ since both its endpoints are open). Now, consider a shortest path $P$ between $v_p$ and $v_q$ in $\widehat{G}_2$. It is obvious that $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q) \leq w(v_p, v_q)$ since $(v_p, v_q)$ itself is a valid path from $v_p$ to $v_q$ in $\widehat{G}_2$. So, it suffices to show

$$\mathtt{dist}_{\widehat{G}_2}(v_p, v_q) \geq w(v_p, v_q), \tag{15}$$

to complete the proof. By the induction hypothesis, we know that $\mathtt{dist}_{\widehat{G}_1}(v_p, v_q) = w(v_p, v_q)$. We show that there exists a path $\tilde{P}$ between $v_p$ and $v_q$ in $\widehat{G}_1$ whose weight is exactly equal to $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q)$. This implies Equation (15) since $w(v_p, v_q) = \mathtt{dist}_{\widehat{G}_1}(v_p, v_q) \leq \mathtt{dist}_{\widehat{G}_2}(v_p, v_q)$. Note that $\tilde{P}$ does not need to be a valid path in $\widehat{G}_2$, the only condition is that the length of $\tilde{P}$ in $\widehat{G}_1$ should be equal to $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q)$ (the length of $P$).

If $P$ does not include the new edge $(v_x, v_y)$, then $\tilde{P} = P$ is also a valid path in $\widehat{G}_1$, and we are done. If $P$ contains the new edge $(v_x, v_y)$, we can exchange this edge $(v_x, v_y)$ with the shortest path between $v_x$ and $v_y$ in $\widehat{G}_1$ (which has weight $w(v_x, v_y)$ by the way this weight is constructed in response to the query $\langle x, y \rangle$). This gives us another path $P'$ between $v_p$ and $v_q$ in $\widehat{G}_1$ whose weight is exactly equal to $\mathtt{dist}_{\widehat{G}_2}(v_p, v_q)$.

## B.5   Proof of Claim B.3

Before we prove Claim B.3, we need the following claim.

**Claim B.7.** *For each edge $(v_p, v_q)$ in $G_f$ such that $w(v_p, v_q) \leq \log_M n$, there exist a path of weight $w(v_p, v_q)$ between $v_p$ and $v_q$ in $G_f$ consisting only of edges of weight 1.*

*Proof.* We prove this on the current graph by induction on the number of queries. Initially, there is no edge between any $v_p$ and $v_q$, so there is nothing to prove. Now, assume the lemma is correct for a graph $G_1$ and consider a new query $\langle p, q \rangle$. If the adversary reports the distance of $p$ and $q$ as 1, then the lemma is obvious for the new edge $(v_p, v_q)$. Otherwise, $w(v_p, v_q) = \mathtt{dist}_{\widehat{G}_1}(v_p, v_q)$. Let $G_2$ be the new graph and $P$ be one shortest path between $v_p$ and $v_q$ in $\widehat{G}_1$. This path $P$ contains at most one edge (say $e$) of weight 1 between two open nodes and all of the other edges are present in $G_1$. Note that none of these edges are incident to $g^\star$ (since we assumed $w(v_p, v_q) \leq \log_M n$), which means we can use the induction hypothesis on these edges (except for $e$). For each edge $e_i \in P$ (different from $e$) of weight $w(e_i)$, by the induction hypothesis, there is a path $P_i$ of weight $w(e_i)$ consisting only of edges of weight 1 in $G_1$. Finally, all of these paths $P_i$ are present in $G_2$ as well. Also note that the edge $e$ (if it exists) is going to be added to $G_1$ by the adversary, so $G_2$ contains $e$ itself. Now, we can concatenate the $P_i$ (and $e$ if exists) to get a path of weight $w(v_p, v_q)$ consisting only of edges of weight 1 in $G_2$. So, the claim holds for the updated graph. $\square$

Now, we proceed with the proof of Claim B.3. First, we show the claim for $i = 1$.

**Case $i = 1$.** We show this case for a general $z$, not only those points that are contained in the solution $S$. So, in this case, we consider $z$ to be an arbitrary point in the space. If $v_z$ is open, by the definition, it is obvious that $v_z$ has at most $M$ neighbors in $G_f$, and trivially the distance of $v_z$ to non-neighbor points is at least 2. Now, assume $v_z$ is closed. Consider the last time that $v_z$ was open. So, after handling the next query, $v_z$ becomes closed. Let $G_1$ be the graph maintained by the adversary just before handling this query. Since $v_z$ is open in $G_1$, the degree of $v_z$ is at most

$M - 1$. In the next step, at most two edges are added to $G_1$, and $v_z$ becomes closed. So, the degree of $v_z$ is at most $M + 1$. One of the neighbors of $v_z$ is $g^\star$. There are at most $M$ other neighbors which we denote by set $\mathcal{N}$. Note that there is no edge between $v_z$ and any other nodes outside $\mathcal{N} + g^\star$. After this time, since $v_z$ is closed, the distance between $v_z$ and any node $v_q$ outside $\mathcal{N} + g^\star$ is going to be at least 2. The reason is that the weight of the shortest path between $v_z$ and $v_q$ in $\widehat{G}$ that adversary considers (any time afterward) is at least 2 (there is no edge of weight 1 between $v_z$ and $v_q$). As a result, the only nodes that might have distance 1 to $v_z$ are in $\mathcal{N}$. Since $|\mathcal{N}| \leq M$, we are done.

**General** $1 \leq i \leq \log_M n$**.** Consider $G_f$. Since the adversary queried the distance from $z$ to every other point, we know that for each node $v_p$ in the graph $(v_z, v_p)$ is an existing edge in $G_f$. Assume the distance between $v_z$ and $v_p$ in the final metric is $i$. According to Lemma B.2, this distance equals $w(v_z, v_p)$ and according to Claim B.7 (since $w(v_z, v_p) = i \leq \log_M n$), there is a path $v_z = v_{p_0}, v_{p_1}, \ldots, v_{p_i} = v_p$ between $v_z$ and $v_p$ consisting of edges of weight 1. Note that nodes of the path are distinct since this is the shortest path between $v_z$ and $v_p$. As a result, each $v_p$ corresponds to a sequence of $i + 1$ pairwise distinct nodes $v_{p_0}, v_{p_1}, \ldots, v_{p_i}$, such that the weight of the edge between each two consecutive nodes is 1. The number of such sequences is at most $M \cdot (M - 1)^{i-1}$. This is because, we have one choice for $v_{p_0}$ which is $v_z$, and $M$ choices for $v_{p_1}$ where there is an edge of weight 1 between $v_{p_0}$ and $v_{p_1}$ (according to the proof of the above case $i = 1$). Then, for each $j \geq 2$, we have at most $M - 1$ options for $v_{p_j}$ since there should exist an edge of weight 1 between $v_{p_{j-1}}$ and $v_{p_j}$, and also $v_{p_j}$ should be different from $v_{p_{j-2}}$. This completes the proof.

# C    Our Results for Deterministic $k$-Means

In this section, we briefly describe our results for the $k$-means problem, where the clustering objective is $\sum_{x \in V} w(x) \cdot d(x, S)^2$ instead of $\sum_{x \in V} w(x) \cdot d(x, S)$.

## C.1    Our Deterministic Algorithm for $k$-Means

We first note that our deterministic algorithm with near-optimal query complexity (see Section 3.4) extends seamlessly to the $k$-means problem, giving us the following result.

**Theorem C.1.** *There is a deterministic algorithm for $k$-means that, given a metric space of size $n$, computes a $O(\log^2(n/k))$-approximation with $\tilde{O}(nk)$ queries (but exponential running time).*

*Proof.* Following the same approach as Appendix A.4, consider the *normalized $k$-means* objective $\mathsf{cost}_2(S) := \left( \sum_{x \in V} w(x) \cdot d(x, S)^2 \right)^{1/2}$. Since the projection lemma (Lemma 2.1) holds for this objective, the algorithm and analysis extend immediately to the normalized $k$-means problem. By observing that a solution $S$ is an $\alpha^2$-approximation to $k$-means if and only if it is an $\alpha$-approximation to normalized $k$-means, it follows that the algorithm produces a $O(\log^2(n/k))$-approximation for the $k$-means problem. $\qquad\square$

Unfortunately, it is not clear how to extend our efficient algorithm from Section 5 to $k$-means. The barrier is that we don't know how to design a good deterministic algorithm for the restricted $k$-means (or normalized $k$-means) problem. In particular, it is not known whether or not the reverse greedy algorithm can be used to produce good solutions for the $k$-means problem. If we want to extend the analysis to the $k$-means objective, the analysis fails since we need the projection lemma to hold. On the other hand, if we want to extend the analysis to the normalized $k$-means objective (where the projection lemma does hold), the failure point of the analysis is Claim 4.3.

## C.2  Our Lower Bound for Deterministic $k$-Means

Our lower bound for deterministic $k$-median (Theorem B.1) extends immediately to deterministic $k$-means. In particular, we get the following theorem.

**Theorem C.2.** *For every $\delta \geq 1$, any deterministic algorithm for the $k$-means problem that has a running time of $O(kn\delta)$ on a metric space of size $n$ has an approximation ratio of*

$$\Omega\left(\left(\frac{\log n}{\log\log n + \log k + \log \delta}\right)^2\right).$$

*Proof.* By changing the values of the parameters in the analysis for $k$-median, we can obtain our lower bound for $k$-means. Let $M = 10k\delta \log^2 n$. Then the cost of the solution $S$ returned by the algorithm is at least $(1 - 2k/M) \cdot n \cdot r^2 \geq (n/2) \cdot r^2$, where $r = \lfloor \log_M n \rfloor$. This follows from the same argument as the proof of Lemma B.4, except that using the $k$-means objective instead of the $k$-median objective gives us the $r^2$ term. By the same argument as the proof of Lemma B.6, the cost of the optimum solution is at most

$$(10k\delta/M) \cdot (2\log_M n)^2 + (1 - 10k\delta/M) \cdot n \cdot 1 = 4 \cdot \frac{\log_M^2 n}{\log^2 n} + n \leq 5n.$$

Thus, the approximation ratio of the algorithm is at least

$$\frac{(n/2)\lfloor \log_M n \rfloor^2}{5n} = \Omega(\log_M^2 n) = \Omega\left(\left(\frac{\log n}{\log\log n + \log k + \log \delta}\right)^2\right).^{[16]} \qquad \square$$

---

[16]Similar to $k$-median, we assume that $M \leq n$. Otherwise, the theorem is trivial.