# Metric $k$-Median Clustering in Insertion-Only Streams

Vladimir Braverman[a,1], Harry Lang[c,2], Keith Levin[b,3,*], Yevgeniy Rudoy[d]

[a] *Johns Hopkins University Department of Computer Science, Baltimore, MD USA*
[b] *University of Wisconsin-Madison Department of Statistics, Madison, WI USA*
[c] *Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, Cambridge, MA USA*
[d] *Johns Hopkins University Department of Applied Mathematics and Statistics, Baltimore, MD USA*

**Abstract**

We present a low-constant approximation for the metric $k$-median problem on insertion-only streams using $O(\epsilon^{-3} k \log n)$ space. In particular, we present a streaming $(O(\epsilon^{-3} k \log n), 2+\epsilon)$-bicriterion solution that reports cluster weights. Running the offline approximation algorithm due to [1] on this bicriterion solution yields a $(17.66+\epsilon)$-approximation [2, 3, 4]. Our result matches the best-known space requirements for streaming $k$-median clustering while significantly improving the approximation accuracy. We also provide a lower bound, showing that any polylog$(n)$-space streaming algorithm that maintains an $(\alpha, \beta)$-bicriterion must have $\beta \geq 2$. Our technique breaks the stream into segments defined by jumps in the optimal clustering cost, which increases monotonically as the stream progresses. By storing an accurate summary of recent segments of the stream and a lower-space summary of older segments, our algorithm maintains a $(O(\epsilon^{-3} k \log n), 2+\epsilon)$-bicriterion solution for the entirety of the stream.

In addition to our main result, we introduce a novel construction that we call a *candidate set*. This is a collection of points that, with high probability, contains $k$ points that yield a near-optimal $k$-median cost. We present an algorithm called monotone faraway sampling (MFS) for constructing a candidate set in a single pass over a data stream. We show that using this candidate set in tandem with a coreset speeds up the search for a solution set of $k$ cluster centers upon termination of the data stream. While coresets of smaller asymptotic size are known, comparative simplicity of MFS makes it appealing as a practical technique.

*Keywords:* Streaming algorithms, $k$-median, clustering

*Corresponding author
*Email address:* `kdlevin@wisc.edu` (Keith Levin)

[1] Supported in part by the National Science Foundation under Grants IIS-1447639 and CCF-1650041.
[2] Supported by the Franco-American Fulbright Commission. The author thanks INRIA (l'Institut national de recherche en informatique et en automatique) for hosting him during the writing of this paper.
[3] Supported by the University of Wisconsin-Madison, Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation.

## 1. Introduction

The metric $k$-median clustering problem is to identify, given a collection of points in a metric space $(\mathcal{X}, d)$, a set of $k$ points, called centers, that (approximately) minimize the sum of the distances of each point in the collection to its nearest center. We consider this problem in the insertion-only streaming setting, in which a collection of $n$ points from $\mathcal{X}$ are presented sequentially in a data stream $D = p_1, p_2, \ldots p_n$, and we are tasked with identifying a clustering solution (i.e., producing a set of $k$ centers) based on a single pass through the stream of points, ideally using memory that grows sublinearly in $n$. That is, our goal is to find a collection $C \subseteq \mathcal{X}$ of cardinality $k$ that minimizes

$$\sum_{i=1}^{n} \min_{c \in C} d(p_i, c),$$

while using storage that grows sublinearly, and ideally polylogarithmically, in $n$.

Streaming clustering dates to the work of Guha, Meyerson, Mishra, Motwani and O'Callaghan [2]. There have been two main classes of algorithms for solving the streaming version of metric $k$-medians in polylogarithmic space complexity. The first class are those based on facility location algorithms, dating to the first polylogarithmic solution for the streaming $k$-median problem due to Charikar, O'Callaghan, and Panigrahy [3]. These methods build upon the connection between the $k$-median problem and the facility location problem, using as a subroutine the online facility location algorithm of Meyerson [5]. Facility-based algorithms achieve low space complexity (e.g., $O(k \log n)$-space due to [4]), but tend to suffer from a comparatively large approximation ratio. The best-space algorithm in this class provides a $(O(k \log n), 1063)$-bicriterion solution (see Section 3.6 for a calculation of this constant). The second class of streaming $k$-median algorithms are those based on coresets [6, 7, 8, 9, 10], by which we mean algorithms that can return a $(k, \epsilon)$-coreset for any $\epsilon > 0$. These algorithms achieve an arbitrarily low approximation ratio, at the expense of significantly more storage, the lowest at the time of this writing being a $O(\epsilon^{-2} k \log^4 n)$-space coreset due to [8]. In these approaches, one typically starts with an offline coreset construction procedure, which is then transformed into a streaming construction using the merge-and-reduce technique of [11]. The cost of using the merge-and-reduce technique is to multiply the space complexity by a factor of $\Omega(\log^3 n)$. Although other methods have been found for the Euclidean case [12, 13], this remains the only technique available for constructing coresets in general metric spaces. Without overcoming this nearly 40-year-old barrier, coreset-based algorithms cannot match the space complexity of facility-based algorithms.

These two classes of algorithms suggest a trade-off between the favorable space-bounds of facility-based methods and the favorable approximation ratio of coreset-based methods. A natural question concerns whether or not it is possible to design an algorithm that performs well in both space and approximation ratio. For Euclidean space, this question was answered in the affirmative by [14]. In 2009, Guha [15] introduced a facility-based $(34 + \epsilon)$-approximation for metric spaces requiring memory of size $O(\epsilon^{-3} \log \frac{1}{\epsilon} k \log^2 n)$. This result straddles the above-mentioned space-accuracy trade-off by offering a low-constant (although not as low as that achieved by coreset-based algorithms) as well as low-space (although not as low as the $O(k \log n)$ achieved by the best facility-based algorithms).

## 2. Our Contribution

Our main result, stated in Theorem 11, is an algorithm that maintains a $(k, 2+\epsilon)$-bicriterion for the streaming metric $k$-medians problem using $O(\epsilon^{-3} k \log n)$ space. This in turn yields an approximate solution to the streaming metric $k$-median problem with space requirements and approximation ratio that are both better than those in [15], using a technique entirely different from that of [14]. Our algorithm requires $O(\epsilon^{-3} k \log n)$-space to maintain a $(O(\epsilon^{-3} k \log n), 2+\epsilon)$-bicriterion solution. It is a well-known result [4, 3, 2, 15] that running an offline $\gamma$-approximate clustering algorithm on an $(\alpha, \beta)$-bicriterion solution yields a $(\beta + 2\gamma(1+\beta))$-approximate clustering. With our $(O(\epsilon^{-3} k \log n), 2+\epsilon)$-bicriterion solution, running the offline 2.61-approximation due to [1] thus yields a $(17.66 + \epsilon)$-approximate solution to the streaming $k$-median problem. This improves the approximation factor of [15] by almost half, and the space requirement is improved from $O(k \log^2 n)$ to $O(k \log n)$. We show in Section 3.5 that this result is essentially optimal, in the sense that no polylog$(n)$-space algorithm can improve upon the approximation ratio achieved by our algorithm.

Our algorithm works in three "layers" running in parallel. Upon the arrival of a new point from the stream, each of these three layers are updated. The first layer is a black-box $O(1)$-approximation for the clustering problem. This approximation algorithm simply runs in the background while the higher layers save information from it. The second layer (see Section 3.2) maintains a summary of a prefix of the stream that is guaranteed to contribute at most an $\epsilon$-small portion to the optimal clustering cost of the stream up to the current time. At any given time, this layer only requires space to store the output of the first layer produced at two previous times. The third layer (see Section 3.3) runs, in parallel, multiple instances of the facility location algorithm due to [5]. These three layers of computation jointly ensure that we maintain a solution throughout the stream that achieves a good approximation to the optimal clustering solution with high probability.

Our techniques differ from previous facility-based algorithms in crucial ways. Like the previous works of [4, 3], our algorithm operates by breaking the stream up into subsequences of points called *phases*. The techniques used in these prior works tend to incur additional cost with each additional phase (i.e., as more points arrive in the stream). In contrast, we avoid additional costs by maintaining a summary of a prefix of the stream while ensuring that the cost of clustering the prefix accounts for only a small fraction of the total cost of clustering the stream. This requires that we have an $O(1)$-approximate estimate of the optimal cost of clustering the stream, without having observed it in its entirety. Of course, such an approximation is impossible to guarantee. Instead, our Algorithm 1 allows us to pretend that we have such an estimate. The fundamental idea is to always maintain a summary of the "next prefix" of the stream. If we detect that the optimal clustering cost of the stream may have exceeded our current upper bound, we replace our current prefix summary with the new one and update our upper bound estimate accordingly. Meanwhile, given an $O(1)$-approximate estimate of the optimal clustering cost, we can construct a good approximation for the stream observed since the end of the prefix (see Section 3.3). Combining both these pieces, we are able to maintain a low-constant solution over the stream.

In addition to our main theoretical result, we also present in Section 4 a technique for constructing what we term a *candidate set*, a data structure that complements the coreset-based techniques discussed above. The motivation for this construction is the *approximate cost oracle* property of coresets. By this we mean the property whereby the cost of $k$-median clustering on a coreset using a given set of $k$ centers has cost that is (up to an additive error) equal to the cost of clustering the entire data stream with those $k$ centers. When using a coreset, upon termination of the stream, one must perform an offline clustering of the coreset in order to find a clustering solution. This offline clustering procedure may be computationally intensive, owing to the need to entertain all points in the coreset as potential cluster centers. The idea behind a candidate set is to avoid this by instead constructing, in parallel to the coreset, a set of points $M$ with cardinality significantly smaller than the coreset, with the guarantee that $M$ contains a set of $k$ points that achieves near-optimal clustering cost when applied to the data stream. Upon termination of the stream, one may then find a solution more quickly by searching only over the centers from the candidate set $M$, using, for example, the local search technique due to [16]. Thus we reap the storage advantages of coresets while avoiding the computational cost of searching over all possible sets of $k$ centers from the coreset. While such approaches fall short of being competitive with the best known approaches for insertion-only streaming $k$-median clustering, the comparative simplicity of the MFS algorithm makes it an appealing candidate for use in practical applications compared with the best known theoretical algorithms, and may be of independent interest as a framework to be applied to other streaming problems.

## 3. Phase-Based Streaming *k*-Median Clustering

In this section, we present our main result, a phase-based algorithm for $k$-median clustering in insertion-only streams. Our algorithm maintains a $(k, 2+\epsilon)$-bicriterion solution using $O(\epsilon^{-3} k \log n)$ space. The algorithm has three major components, which we conceptualize as operating in three sequential layers. The first is a black-box $O(1)$-approximation; a single instance simply runs in the background while the higher layers save information from it. The second layer, presented in Section 3.2, maintains a summary of the prefix of the stream that contributes at most an $\epsilon$-small portion to the total clustering cost of the stream. Maintaining this prefix requires only enough space to store the output of the black-box constant approximation running in the first layer. The third layer, described in Section 3.3, runs parallel instances of the facility location algorithm of [5] in phases, adjusting the facility cost as the stream progresses. Four parallel instances in this layer (two in each of two different phases at any given time) are sufficient to maintain the $1 - \frac{1}{n}$ probability of success attained by most approximate streaming clustering algorithms, but this probability can be amplified by increasing the number of parallel instances.

### *3.1. Definitions and Notation*

Before proceeding, we pause to establish notation and definitions. We assume that we are given a data stream $D = p_1, p_2, \ldots, p_n$, consisting of a sequence of points lying in some metric space $(\mathcal{X}, d)$. We define the distance of a point $p \in \mathcal{X}$ to a set $\Phi \subseteq \mathcal{X}$ as $d(p, \Phi) = \min_{\phi \in \Phi} d(p, \phi)$, with $d(p, \emptyset) = \infty$ by convention. For $1 \leq i \leq j \leq n$, we write $[i, j]$ to denote the subsequence of the stream given by $p_i, p_{i+1}, \ldots, p_j$, so that, for example, $[1, N]$ denotes the first $N$ points appearing in the stream. We assume that each point $p$ appearing in the stream has an associated integral weight $w(p) \in \{0, 1, 2, \ldots\}$. We let $n$ denote the total weight of the stream, $n = \sum_{p \in D} w(p)$, while noting that we may equivalently consider $D$ to be a sequence of points of unit weight, and represent a weight-$w$ point in our original weighted stream by repeating the same point $w$ times. In light of this fact, we will simply use $n$ to denote the length of the stream throughout this work, with no real loss of generality. We assume throughout that $n$ is known in advance, though this is for ease of exposition— a polynomial upper-bound on $n$ is sufficient for our purposes. We note that $n$ is assumed to be known in advance in most recent works on this problem [3, 4, 14], and removing this assumption is non-trivial. For a set (or multiset) $A$, we denote its cardinality by $|A|$.

The goal of clustering is to identify a set of points that minimizes a clustering objective function, in which we pay a cost to connect each point in the data set to one of $k$ cluster centers. In the present work, we are concerned with the $k$-median clustering objective.

**Definition 1** (Cost Function)**.** *Given a multiset $A \subset \mathcal{X}$ and a set $C \subset \mathcal{X}$, the function $\nu(A, C)$ denotes the cost of clustering $A$ with a set of cluster centers $C$, defined to be*

$$\nu(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c).$$

**Definition 2** (Optimum Cost)**.** *For a particular multiset of points $A \subset \mathcal{X}$ and a set $B \subset \mathcal{X}$, we denote by $\nu_{\mathrm{OPT}}(A, B, k)$ the lowest possible cost of clustering $A$ with $k$ centers from $B$. That is, $\nu_{\mathrm{OPT}}(A, B, k) = \min_{C \in B^k} \nu(A, C)$. When the set $B$ is known from context (e.g., if $B$ is the entire metric space $\mathcal{X}$ or $B$ is the set of all points appearing in the data stream $D$), we will write simply $\nu_{\mathrm{OPT}}(A, k)$.*

**Definition 3** (Connect Function)**.** *Let $A, B$ be multisets of equal cardinality. We write* Connect$(A, B)$ *to denote the minimum connection cost of transporting each $a \in A$ to a unique $b \in B$. That is,*

$$\mathrm{Connect}(A, B) = \min_t \sum_{a \in A} d(a, t(a)),$$

*where the minimization is over all bijective maps $t : A \to B$.*

Before proceeding, we make two observations that will prove useful in Sections 3.2 and 3.4, respectively.

**Observation 4.** *Let $A$ and $B$ be multisets of equal cardinality with elements drawn from the metric space $(\mathcal{X}, d)$ and let $C \subset \mathcal{X}$. Then*

$$\nu(A, C) \leq \text{Connect}(A, B) + \nu(B, C).$$

*Proof.* Let $g$ be the function that assigns each element of $B$ to its nearest point in $C$. That is, the map from $B$ to $C$ that attains cost $\nu(B, C)$. Let $t$ be the optimal bijective map from $A$ to $B$ that attains $\text{Connect}(A, B)$. By the triangle inequality, for every $a \in A$, $d(a, g(t(a))) \leq d(a, t(a)) + d(t(a), g(t(a)))$. Let $h$ be the optimal map assigning each $a \in A$ to its nearest point in $C$. The result follows by summing over all $a \in A$ and noting that $d(a, h(a)) \leq d(a, g(t(a)))$. □

**Observation 5.** *Let $A_1, B_1, A_2$ and $B_2$ be multisets of elements from the metric space $(\mathcal{X}, d)$ with $|A_1| = |B_1|$ and $|A_2| = |B_2|$. If $\text{Connect}(A_1, B_1) \leq v_1$ and $\text{Connect}(A_2, B_2) \leq v_2$, then $\text{Connect}(A_1 \cup A_2, B_1 \cup B_2) \leq v_1 + v_2$.*

*Proof.* Let $t_i$ be the optimal bijective map from $A_i$ to $B_i$. Then consider $g(a) = t_i(a)$ if $a \in A_i$. While $g$ may not be the optimal bijective map from $A_1 \cup A_2$ to $B_1 \cup B_2$, it yields an upper bound, completing the proof. □

### *3.2. Phase Manager*

As mentioned above, our algorithm relies on maintaining an estimate of the optimal $k$-median clustering cost of the stream to within a multiplicative error (refer to Theorem 9 below to see where this estimate is used). Recalling our notation $[1, N]$ to denote the first $N$ points of the stream, we require a way to maintain a monotonically increasing function $f([1, N])$ such that for all $N$,

$$\nu_{\text{OPT}}([1, N], \mathcal{X}, k) \leq f([1, N]) \leq \theta \nu_{\text{OPT}}([1, N], \mathcal{X}, k) \tag{1}$$

for a constant $\theta$ that we will specify below. Note that here and in the sequel, we will suppress the metric space $\mathcal{X}$ from our notation $\nu_{\text{OPT}}([1, N], \mathcal{X}, k)$, taking this metric space to be understood, and instead simply write $\nu_{\text{OPT}}([1, N], k)$.

Operating on an insertion-only stream $D$, Braverman, et al. [4] introduced a modification of the PLS algorithm initially presented in [3], which we refer to as $\text{PLS}_+$. $\text{PLS}_+$ maintains a multiset $Q$ such that for all $N$, $\text{Connect}([1, N], Q) \leq \mu \nu_{\text{OPT}}([1, N], k)$ for some constant $\mu$. $\text{PLS}_+$ constructs $Q$ according to a technique that connects each arriving point in $D$ to (weighted) points seen previously in the stream. We make a simple modification to maintain, in addition, a value $q$ such that $\text{Connect}([1, N], Q) \leq q$. For a portion $P$ of the stream, we write $\text{PLS}_+(P)$ to denote the multiset $Q$ built on this portion of the stream, and we denote the resultant upper bound on Connect$(P, Q)$ by $q(P)$. Via a standard argument (see, e.g., [3]), running an offline $\gamma$-approximation for $k$-median on $Q$ yields a $2\gamma(1+\mu)$-approximate solution on the original input. Thus, we will see below that we can take the constant $\theta$ in Equation (1) to be

$$\theta = 2\gamma(1+\mu). \tag{2}$$

Beyond simply computing a clustering solution, we use the output of the $\text{PLS}_+$ algorithm to build a monotonically increasing function $f$ satisfying Equation (1) as follows. Define $f'$ to be the sum of $q([1, N])$ and the cost of clustering $\text{PLS}_+([1, N])$ with its $\gamma$-approximate $k$-median clustering. By Observation 4, $f'$ satisfies the desired inequalities in Equation (1), but $f'$ may not be monotonically increasing, as the clustering cost of the $\gamma$-approximate solution need not be monotonic in the size of the input. To get around this, we define $f$ recursively according to $f([1, N]) = \max\{f'([1, N], f([1, N-1])\}$. Upon arrival of a new point $p_N$, having already computed $f([1, N-1])$, we obtain $f'([1, N])$ by adding $p_N$ to the input to our running instance of $\text{PLS}_+$ and using the new $\text{PLS}_+$ cost $q([1, N])$. Thus, computing $f([1, N])$ requires $O(1)$ additional computation time and storage beyond the cost of running $\text{PLS}_+$ on the stream. By construction, $f$ is guaranteed to be monotonically increasing. Moreover, it still satisfies the desired inequalities in Equation (1) because $\nu_{\text{OPT}}([1, N], k)$ is monotonically increasing in $N$.

The purpose of the second layer of our algorithm is to keep track of a prefix of the stream such that at any given time $N$, the connection cost of that prefix accounts for at most a small fraction of the total optimal clustering cost $\nu_{\text{OPT}}([1, N], k)$. This is achieved by maintaining a two-level filtration of the stream. Having observed the first $N$ points of the stream $[1, N] = p_1, p_2, \ldots, p_N$, we denote the elements of this filtration by $A_N$ and $B_N$, where $\emptyset \subseteq A_N \subseteq B_N \subseteq [1, N]$. Specifically, we take $A_N$ and $B_N$ to be prefixes of the stream, meaning that they are each equal to some sequence $[1, m]$ for some $1 \le m \le N$. Further, we select these prefixed in such a way that the following two loop invariants are maintained:

$$\begin{aligned} f(A_N) &\le \frac{\epsilon}{\mu\theta} f(B_N) \\ f(B_N) &> \frac{\epsilon}{\mu\theta} f([1, N]), \end{aligned} \tag{3}$$

where $\mu$ is the approximation constant guaranteed by $\text{PLS}_+$, $\theta$ is as in Equation (2), and $f$ is the function constructed by the first layer to satisfy Equation 1.

At the beginning of the stream, it will be necessary to establish these two loop invariants. Let $m$ be the smallest number such that $\{p_1, p_2, \ldots, p_m\}$ contain exactly $k+1$ distinct points. We define $A_t$ and $B_t$ arbitrarily for $t < m$, noting that solving the $k$-medians clustering problem on a multiset of at most $k$ distinct points can be solved trivially by taking the centers to be equal to those points. We take $A_m$ to be the empty set and $B_m$ to be the first $k+1$ distinct points in the stream. We observe also that even if $k+1$ distinct points do not arrive until $m$ is much greater than $k+1$, this initialization procedure can be performed in $O(k \log n)$.

Having established the loop invariants in Equation (3), Algorithm 1 maintains these invariants with a single instance of $\text{PLS}_+$. When a point arrives, it simply updates the running instance of $\text{PLS}_+$ and, if necessary, updates $A_N$ and $B_N$ in terms of $A_{N-1}$ and $B_{N-1}$ in order to maintain the loop invariants. Specifically, if the condition in Line 2 is satisfies, this indicates that the second loop invariant in Equation (3) has been violated, and we need to discard our previous prefix $A_{N-1}$ and update our stream prefix to be the old $B_{N-1}$. When this occurs, we say that we have begun a new *phase*. Note that Algorithm 1 does not store any information besides the state of $\text{PLS}_+$ for each element of the current filtration. Therefore, the memory requirement for this portion of our algorithm is precisely that required to run an instance of $\text{PLS}_+$.

**Algorithm 1** Update Process, upon arrival of point $p_N$. $\epsilon > 0$ is a user-specified approximation constant. $\mu$ is the constant of approximation guaranteed by $\text{PLS}_+$. $\theta$ is as defined in Equation (2).

1: Update $\text{PLS}_+$ with $p_N$ and compute $f([1, N])$
2: **if** $f([1, N]) \geq \frac{\mu\theta}{\epsilon} f(B_{N-1})$ **then**
3: $\quad A_N \leftarrow B_{N-1}$; $B_N \leftarrow [1, N]$
4: **else**
5: $\quad A_N \leftarrow A_{N-1}$; $B_N \leftarrow B_{N-1}$
6: **end if**

**Lemma 6.** *Suppose that $\mu\theta/\epsilon > 1$ in Algorithm 1. Using $O(k \log n)$ memory, Algorithm 1 maintains a filtration $\emptyset \subset A_N \subset B_N \subset [1, N]$ such that for all $N$, the invariants in Equation* (3) *are satisfied. That is, for all $N$,*

$$f(A_N) \leq \frac{\epsilon}{\mu\theta} f(B_N) \quad \text{and} \quad f(B_N) > \frac{\epsilon}{\mu\theta} f([1, N]).$$

*Proof.* If the condition on Line 2 is not satisfied, then both invariants in Equation (3) continue to hold, and we can simply leave our filtration unchanged. If the condition on Line 2 is satisfied, then the second invariant in Equation (3) has been violated, and must be re-established in Line 3. We recursively assume that both invariants held for the filtration of $[1, N-1]$. For ease of notation, write $\beta = \mu\theta/\epsilon$. The first invariant in Algorithm 1 reads $f(A_N) \leq \beta^{-1} f(B_N)$, which is equivalent to $f(B_{N-1}) \leq \beta^{-1} f([1, N-1])$. This is guaranteed to hold, since the second invariant was violated by assumption. The second invariant reads $f(B_N) > \beta^{-1} f([1, N])$. Since we have $B_N = [1, N]$ on Line 3 and $\beta > 1$, Line 3 reestablishes the second invariant. □

Algorithm 1 guarantees that when a phase change occurs, $\nu_{\text{OPT}}(D, k)$ will remain within a constant multiplicative range of our present (estimated) upper bound until the next phase change, as the following result shows.

**Lemma 7.** *Let $\mu$ be the constant of approximation guaranteed by* $\text{PLS}_+$ *and let $\theta$ be as in Equation* (2). *Provided that $\epsilon > 0$ satisfies $\mu\theta/\epsilon > 1$, Algorithm 1 guarantees that for all $N$,*

$$\frac{f(B_N)}{\theta} \leq \nu_{\text{OPT}}([1, N], k) < \frac{\mu\theta}{\epsilon} f(B_N).$$

*Proof.* The first inequality follows from the approximation guarantee of $f$ and monotonicity. The second inequality follows from the second loop invariant of Algorithm 1, which is ensured by Lemma 6, and the fact that $\nu_{\text{OPT}}([1, N], k) \leq f([1, N])$. □

As discussed above, the goal of maintaining a prefix $A_N$ (and a "next" prefix $B_N$) is to ensure that at any time, we can safely assume that the cost of clustering the prefix contributes only an $\epsilon$-small fraction of the total optimal clustering cost. The following lemma makes this precise.

**Lemma 8.** *Provided that $0 < \epsilon \leq 1/2$ and $\mu\theta/\epsilon > 1$, at all times $N$, the prefix $A_N$ maintained by Algorithm 1 is such that*

$$\text{Connect}(A_N, \text{PLS}_+(A_N)) \leq \epsilon\nu_{\text{OPT}}([1, N], k).$$

*Proof.* By Lemma 6, the function $f$ based on Algorithm 1 guarantees that $f(A_N) \leq \epsilon f(B_N)/\mu\theta$. Using the fact that $f(A_N)$ is a trivial upper bound on $\nu_{\text{OPT}}(A_N, k)$ and applying Lemma 7, we have

$$\nu_{\text{OPT}}(A_N, k) \leq f(A_N) \leq \frac{\epsilon}{\mu\theta} f(B_N) \leq \frac{\epsilon}{\mu} \nu_{\text{OPT}}([1, N], k).$$

As discussed above, the $\text{PLS}_+$ algorithm running in the background of Algorithm 1 yields a weighted set $\text{PLS}_+(A_N)$, such that $\text{Connect}(A_N, \text{PLS}_+(A_N)) \leq \mu\nu_{\text{OPT}}(A_N, k)$. Combining this with the above display, we have shown that $\text{Connect}(A_N, \text{PLS}_+(A_N)) \leq \epsilon\nu_{\text{OPT}}([1, N], k)$, completing the proof. □

In the next two sections, we will use the guarantees of Algorithm 1 to construct a $(O(\epsilon^{-3} k \log n), 2 + \epsilon)$-bicriterion clustering solution. A collection of subroutines running in parallel in the third layer of our algorithm will observe (but not influence) Algorithm 1 and store two sets at any given time $N$: $\text{PLS}_+(A_N)$ and $\text{PLS}_+(B_N)$. That is, the third layer stores the state of the $\text{PLS}_+$ algorithm as run on the current prefix $A_N$ and on the "next" prefix $B_N$. In this way, we will maintain a $k$-median clustering solutions on both $A_N$ and $B_N$ at all times $N$.

### *3.3. Facility Manager*

The bookkeeping done by Algorithm 1 ensures that at any time, the cost of clustering the prefix $A_N$ contributes at most an $\epsilon$ fraction of the optimal clustering cost $\nu_{\text{OPT}}([1, N], k)$ (see Lemma 8 below). It remains, then, to ensure that we can produce at any time $N$ a close approximation for clustering the remainder of the stream, $[1, N] \setminus A_N$. Toward this end, in this section we present Algorithm 3, which runs in parallel with Algorithm 1. We make use of a modified version of the online facility location algorithm due to [5] as a subroutine. The main result of this section is Theorem 10, which establishes that Algorithm 3 maintains, at each time $N$, a weighted set $Q_N$ such that with high probability,

$$\text{Connect}([1, N] \setminus A_N, Q_N) \leq (2 + \epsilon)\, \nu_{\text{OPT}}([1, N], k).$$

Toward this end, we recall the OFL Algorithm 2, as used in [3], which solves the facility location problem with facility cost $\kappa$. We use the OFL algorithm to maintain a weighted set of facilities (i.e., points from the stream) $\Phi$. Upon the arrival of a point $p$, we open a weight $w(p)$ facility at point $p$ with probability $w(p)d(p, \Phi)/\kappa$; otherwise we connect it to the nearest open facility, incrementing that facility's weight by $w(p)$ and paying service cost $w(p)d(p, \Phi)$. We will see in Theorem 9 below that by choosing the facility cost $\kappa$ appropriately, we can ensure that $\Phi$ achieves a service cost that is within a constant multiple of the optimal $k$-median clustering cost of the input.

**Algorithm 2** The OFL algorithm. $\kappa$ is a user-supplied facility cost.

```
ServiceCost ← 0; FacilityCount ← 0; Φ ← ∅
Update Process, upon arrival of point p_N:
if a probability min(1, w(p_N)d(p_N, Φ)/κ) event occurs then
    Open a facility at p_N with weight w(p_N)
    FacilityCount ← FacilityCount +1
else
    Increment weight of a nearest facility to p_N by w(p_N)
    ServiceCost ← ServiceCost +w(p_N)d(p_N, Φ)
end if
```

The output from running Algorithm 2 on the stream is a set of points $\Phi$ (i.e., the facilities opened by the algorithm), with each facility $p \in \Phi$ weighted according to the total weight of the points served by $p$. The following theorem shows that the facility set $\Phi$ produced by Algorithm 2 applied to a weighted set of points $A$ has total service cost that is a good approximation to the optimal $k$-median cost $\nu_{\text{OPT}}(A, k)$. Further, we show that the total number of facilities $|\Phi|$ opened by the algorithm is at most a multiplicative factor larger than $k$. Of course, the set $\Phi$ may contain more than $k$ facilities. Below, we will show how to cull the set $\Phi$, which may contain more than $k$ facilities, to obtain a set of $k$ cluster centers. First, we present Theorem 9, which establishes a high-probability bound on the total service cost of the facility set $\Phi$. The result follows by tuning the parameters in Theorem 3.1 of [4]. Our result includes a parameter $\epsilon$, which is set to $\epsilon = 1$ in the original theorem. We thus include a sketch of how to modify the original proof to allow this parameter to vary.

**Theorem 9.** *Suppose that* OFL *runs on a weighted set of points $A$ with total weight at most $n$, using facility cost $\kappa = L/k(1 + \log n)$ where $L \leq \epsilon\nu_{\text{OPT}}(A, k)$, with $\epsilon \in (0, 1/2]$. Then with probability at least $1 - \frac{1}{n}$, at* *most* $7\epsilon^{-1}k(1+\log n)\nu_{\text{OPT}}(A,k)/L$ *facilities are opened, and the total service cost of the resulting solution is at most* $(2+7\epsilon)\,\nu_{\text{OPT}}(A,k)$.

*Proof Sketch.* Consider an optimal center $c$ that services the set $S \subset A$. Let $\sigma = \sum_{p\in S} d(p,c)$ be the total service cost of assigning the points in $S$ to $c$. For $j \geq 0$, define set $S_j \subseteq S$ such that $|S_j| = \epsilon|S|/(1+\epsilon)^j$ and each point in $S_j$ is not farther from $c$ than any point point in $S_{j+1}$. Then for $j' = \log_{1+\epsilon}(n/2) \leq 2\epsilon^{-1}\log n$ and $\epsilon \leq 1/2$, the set $\cup_{j>j'} S_j$ consists of at most a single point. As in the proof of Theorem 3.1 in [4], the service cost of all points after a facility is opened in a region is deterministically at most $(\epsilon/(1-\epsilon)+(1+\epsilon))\sigma$. This follows by applying Markov's inequality to show that the cost of connecting the nearest $\epsilon|S|$ points to the opened facility is at most $\frac{\epsilon}{1-\epsilon}\sigma$. As for before a facility opens, it is shown in [4] that the probability of having total service at least $y$ over $x$ regions before a facility opens is at most $e^{x-y(e-1)/e}$. Setting $x = 2\epsilon^{-1}k(1+\log n)$ and $y = 2e\epsilon^{-1}k(1+\log n)/(e-1)$ yields the result. □

Theorem 9 implies that we can obtain a good approximation to the optimal clustering, so long as the parameter $L$ is within a constant multiple of $\nu_{\text{OPT}}(A,k)$. Algorithm 3 handles the problem of keeping multiple instances of OFL running in parallel while updating this facility cost in response to phase changes identified by Algorithm 1 as the stream proceeds. The algorithm maintains a collection of OFL instances running in parallel. Any time a phase change occurs (i.e., Algorithm 1 updates the filtration $A_N \subseteq B_N$), Algorithm 3 begins running $d+1$ instances of OFL with facility cost set to $\epsilon f(B_N)/\theta$. These instances continue running until the next phase change (Line 1 in Algorithm 3), when we increase the service cost and duplicate the instance $d+1$ times. Thus, each phase over the course of the stream has an associated collection of $d+1$ independent instances of the OFL algorithm. We say that these instances are running in the same *bucket*, with the $t$-th phase having an associated bucket, also numbered $t$. We make the distinction between a phase and a bucket owing to the fact that in Algorithm 3, we will need to initialize the OFL instances running in the $(t+1)$-th bucket upon the start of the $t$-th phase. Note that this is precisely the reason for maintaining the prefix $A_N$ and the "next" prefix $B_N$. Throughout the stream, Algorithm 3 uses the current bucket (i.e., the bucket associated to the current phase) and the next bucket (i.e., the bucket associated to the next phase) to maintain a weighted set of facilities $Q_N$ that is comprised of facilities from two different running OFL instances.

We now present the main theorem of this section.

**Theorem 10.** *Suppose that* $\epsilon \in (0,1/2]$. *With probability at least* $1-n^{-d}$, *where* $d$ *is a chosen parameter, Algorithm 3 maintains a weighted set* $Q_N$ *such that*

$$\text{Connect}([1,N]\setminus A_N, Q_N) \leq (2+7\epsilon)(1+\epsilon)\nu_{\text{OPT}}([1,N],k),$$

*The algorithm requires memory of size* $O(d\epsilon^{-3}k\log n)$.

**Algorithm 3** Update Process, upon arrival of point $p_N$. $\mu$ is the approximation constant guaranteed by $\text{PLS}_+$ and $\theta$ is as defined in Equation (2).

1: **if** $p_N$ causes phase $t$ to begin **then**
2: Terminate all instances in bucket $t-1$
3: Force all instances in bucket $t$ to open $p_N$ as a facility
4: $\Phi_1 \leftarrow$ facilities of a bucket $t$ instance with minimal service cost
5: $\kappa \leftarrow \epsilon f(B_N)/\theta k(1+\log n)$
6: Initialize $d+1$ instances of OFL in bucket $t+1$ with facility cost $\kappa$
7: **else**
8: Update all running instances of OFL with point $p_N$
9: Terminate instances with more than $7\mu\theta^2\epsilon^{-3}k(1+\log n)$ open facilities
10: **end if**
11: **if** bucket $t$ contains a running instance **then**
12: $\Phi_1 \leftarrow$ facilities of an instance in bucket $t$ with minimal service cost
13: **else**
14: $\Phi_2 \leftarrow$ facilities of an instance in bucket $t+1$ with minimal service cost
15: $Q_N \leftarrow \Phi_1 \cup \Phi_2$
16: **end if**

*Proof.* To establish the space bound, note first that Line 9 guarantees that we have at most $7\mu\theta^2\epsilon^{-3}k(1+\log n)$ facilities per instance of OFL. We store at most $d+1$ instances in each of buckets $t$ and $t+1$, resulting in an overall storage of at most $14(d+1)\mu\theta^2\epsilon^{-3}k(1+\log n)$ facilities. The $O(d\epsilon^{-3}k\log n)$ space bound follows since $\mu$ and $\theta$ are assumed constant.

We now prove that with high probability, at least one instance in bucket $t$ remained active throughout phase $t-1$. Specifically, we show that at least one instance in bucket $t$ has not opened too many facilities, thus avoiding termination on line 9. Toward this end, consider the instances in bucket $t$. By definition, these started running as a batch of $d+1$ instances at the beginning of phase $t-1$. Since Algorithm 1 sets $A_N \leftarrow B_{N-1}$ at a phase change, these instances were started with facility cost $\epsilon f(A_N)/\theta k(1+\log n)$ and ran on the segment $B_N \setminus A_N$. We let $B'_N$ denote $B_N$ without the final point that caused the transition to phase $t$, and apply Theorem 9 to $B'_N$. By Lemma 7, we have

$$\frac{\theta \nu_{\text{OPT}}(B'_N, k)}{\epsilon f(A_N)} < \mu\theta^2\epsilon^{-2}.$$

With this bound, Theorem 9 guarantees that with probability $1-n^{-d-1}$, at least one of the $d+1$ instances will open at most $7\mu\theta^2\epsilon^{-2}k(1+\log n)$ facilities when running on $B'_N$. Since the number of facilities opened is monotonically increasing during runtime, this implies the same bound on the number of facilities when running OFL on the segment $B'_N \setminus A_N$. Therefore, with probability $1 - n^{-d-1}$, at least one instance survives to the beginning of phase $t$ by not being terminated on Line 9. At the beginning of phase $t$, we apply the same analysis to $[1, N] \setminus B_N$ (without the need to remove the final point, since the phase has not ended), and arrive at the same probabilistic bound on the number of facilities for instances in bucket $t + 1$.

Let $L_t = k(1 + \log n)\kappa_t$ where $\kappa_t$ is the facility cost used for instances in bucket $t$. We consider two possible cases. In the first case, suppose that $\nu_{\mathrm{OPT}}(B'_N, k) \geq \epsilon\nu_{\mathrm{OPT}}([1, N], k)$. We repeat the previous analysis with Theorem 9 of running OFL on the segment $[1, N] \setminus A_N$ instead of $B'_N \setminus A_N$. Since

$$\frac{\theta\nu_{\mathrm{OPT}}([1, N], k)}{\epsilon f(A_N)} \leq \frac{\epsilon^{-2}\theta\nu_{\mathrm{OPT}}(B'_N, k)}{f(A_N)} < \mu\theta^2\epsilon^{-3}$$

and

$$L_t = \frac{\epsilon f(A_N)}{\theta} \leq \epsilon\nu_{\mathrm{OPT}}(A_N, k) \leq \epsilon\nu_{\mathrm{OPT}}([1, N], k),$$

Theorem 9 gives a high-probability guarantee that at least one OFL instance running in bucket $t$ has opened no more than $7\mu\theta^2\epsilon^{-3}k(1 + \log n)$ facilities with total service cost at most $(2 + 7\epsilon)\nu_{\mathrm{OPT}}([1, N], k)$.

In the second case, suppose that $\nu_{\mathrm{OPT}}(B'_N, k) < \epsilon\nu_{\mathrm{OPT}}([1, N], k)$. If there is an active instance in bucket $t$, then its total connection cost is at most $(2+7\epsilon)\nu_{\mathrm{OPT}}([1, N], k)$. If there are no active instances in bucket $t$, we return $\Phi_1 \cup \Phi_2$. We apply Theorem 9 to $B'_N \setminus A_N$ to show that $\mathrm{Connect}(B'_N \setminus A_N, \Phi_1) \leq (2+7\epsilon)\nu_{\mathrm{OPT}}(B'_N, k)$. Line 3 implies that

$$\mathrm{Connect}(B'_N \setminus A_N, \Phi_1) = \mathrm{Connect}(B_N \setminus A_N, \Phi_1),$$

and therefore $\mathrm{Connect}(B_N \setminus A_N, \Phi_1) < (2 + 7\epsilon)\nu_{\mathrm{OPT}}(B'_N, k)$. By definition,

$$L_{t+1} = \frac{\epsilon f(B_N)}{\theta} \leq \epsilon\nu_{\mathrm{OPT}}(B_N, k) \leq \epsilon\nu_{\mathrm{OPT}}([1, N], k),$$

and we apply the theorem again to $[1, N] \setminus B_N$ to show that $\mathrm{Connect}([1, N] \setminus B_N, \Phi_2) \leq (2+7\epsilon)\nu_{\mathrm{OPT}}([1, N], k)$. Observation 5 then bounds the connection cost as

$$\mathrm{Connect}([1, N] \setminus A_N, \Phi_1 \cup \Phi_2) \leq (2 + 7\epsilon)\left[\nu_{\mathrm{OPT}}(B'_N, k) + \nu_{\mathrm{OPT}}([1, N], k)\right] < (2 + 7\epsilon)(1 + \epsilon)\nu_{\mathrm{OPT}}([1, N], k).$$

Of course, the above proof holds for a single phase. Since there is at least one point per phase, there are at most $n$ phases. Applying a union bound over the phases, the $1 - n^{-d-1}$ probability guarantee for each phase yields a $1 - n^{-d}$ probability guarantee over the full stream, completing the proof. □

From now on, for ease of notation, we refer to the result of Theorem 10 as guaranteeing connection cost $(2 + \epsilon)\nu_{\mathrm{OPT}}([1, N], k)$, rather than $(2 + 7\epsilon)(1 + \epsilon)\nu_{\mathrm{OPT}}([1, N], k)$. This follows by rescaling $\epsilon$ by a suitable constant and making use of the fact that $\epsilon \leq 1/2$ by assumption.

### 3.4. Combining Both Algorithms

The layers of our algorithm described above work in combination to maintain clusterings for two parts of the stream: a prefix $A_N$ and the rest of the stream, $[1,N] \setminus A_N$. Applying the $\text{PLS}_+$ algorithm to this prefix yields a multiset $\text{PLS}_+(A_N)$ such that

$$\text{Connect}\,(A_N, \text{PLS}_+(A_N)) \leq \mu\nu_{\text{OPT}}(A_N, k).$$

Similarly, Algorithm 3 produces a weighted set $Q_N$ such that

$$\text{Connect}([1,N] \setminus A_N, Q_N) \leq (2+\epsilon)\,\nu_{\text{OPT}}([1,N], k).$$

Combining these two outputs, we obtain our desired bicriterion solution for the $k$-median clustering problem.

**Theorem 11.** *With notation as above, define $U_N = \text{PLS}_+(A_N) \cup Q_N$ and suppose that $\epsilon \in (0, 1/2]$ satisfies $\mu\theta/\epsilon > 1$. With high probability, at any time $N$, $U_N$ is a $(O(\epsilon^{-3} k \log n), 2+\epsilon)$-bicriterion solution for $k$-median clustering.*

*Proof.* By Lemma 8, at any time $N$, we have $\text{Connect}(A_N, \text{PLS}_+(A_N)) \leq \epsilon\nu_{\text{OPT}}([1,N], k)$. Similarly, Algorithm 3 provides us with a weighted set $Q_N$ such that $\text{Connect}([1,N] \setminus A_N, Q_N) \leq (2+\epsilon)\,\nu_{\text{OPT}}([1,N], k)$. Define $U_N = \text{PLS}_+(A_N) \cup Q_N$. By Observation 5, $\text{Connect}([1,N], U_N) \leq (2+2\epsilon)\nu_{\text{OPT}}([1,N], k)$. Replacing $\epsilon$ by $\epsilon/2$ yields the desired bicriterion solution. By Theorem 10, the facility manager running in Algorithm 3 maintains $Q_N$ using $O(\epsilon^{-3} k \log n)$ space, while $\text{PLS}_+$ requires $O(k \log n)$ space [4], establishing the desired result. □

### 3.5. Lower Bound

We have shown that the $k$-median clustering problem can be approximately solved using sublinear space, specifically by first producing a bicriterion solution. Here, we establish an accompanying lower bound, showing that no sublinear-space $(\alpha, \beta)$-bicriterion solution is possible for the streaming $k$-median problem for $\beta < 2$. The following lower bound relies on the fact that one can construct problem instances in which certain input points are especially crucial to the clustering cost, but any approximation algorithm using sublinear space must forget most of the input points.

**Theorem 12.** *For the metric $k$-median problem, no streaming algorithm that uses space sublinear in the input size can (with greater than constant probability) maintain a $(\alpha, \beta)$-bicriterion for $\beta < 2$.*

*Proof.* Consider a specific algorithm with space complexity $S(n)$, measured in the number of points able to be stored, and suppose that $S(n) = o(n)$. Define $R(n) = \left\lceil \sqrt{nS(n)} \right\rceil$ and note that $R(n) = o(n)$. We will begin by constructing an input for the 1-median case, and then show that it can be modified for $k$-median. Let the input begin with $(p_1, p_2, \ldots, p_{R(n)})$ with $d(p_i, p_j) = 1$ for all distinct $i, j \in \{1, 2, \ldots, R(n)\}$. That is, the first

$R(n)$ points are, in a certain sense, indistinguishable to any algorithm. Thus, even for a non-deterministic algorithm, there must exist a deterministic $c \in \{1, 2, \ldots, R(n)\}$ such that after the algorithm has seen the first $R(n)$ points of the input, $c$ is stored in memory with probability at most $S(n)/R(n) \le \sqrt{S(n)/n} = o(1)$. Now, construct the entire input to be $(p_1, p_2, \ldots, p_{R(n)}, q_{R(n)+1}, \ldots, q_n)$, where $d(p_c, q_i) = 1 \forall i \in \{R(n)+1, \ldots, n\}$, and set all other distances according to shortest path distance. If our algorithm has failed to store $p_c$ as a potential center, the next best clustering (using one of the first $R(n)$ points as the center) yields a cost of $R(n) - \alpha + 2(n - R(n))$. In contrast, the optimal clustering, which uses $p_c$ as the center, has cost $n - 1$. Since $\alpha$ is a lower-bound on the storage requirement of the algorithm (any algorithm must at least store the solution that it constructs), in the large-$n$ limit, this input yields a cost-ratio approaching 2 with probability approaching 1.

To extend this construction from the 1-median problem to the more general $k$-median setting, use the above input with size $n/k$ and duplicate it $k$ times, where each duplicated set is at least distance 2 from any other. The value of $c$ may be different for each of the $k$ pieces, but there will always exist a deterministic $(c_1, c_2, \ldots, c_k)$ such that the above argument extends straightforwardly. □

### *3.6. Computing the Constant of Previous Algorithms*

We close this section by pausing briefly to compute the constant of the original PLS algorithm presented in [3], which the authors left unspecified. The lower-space algorithm of [4] has an even larger approximation constant, owing to the high-probability guarantee on the facility location lemma of a $(3+\frac{2e}{e-1})$-approximation instead of a 4-approximation. In Section 2 of [3], the connection cost of the maintained PLS set is seen to be $c = 4(1 + 4(a + b))$ for constants $a, b$, which can be freely specified subject to the restraint that $a + 4(1 + 4(a + b)) \le ab$. Minimizing $c$ as a function of $a$ and $b$ subject to this constraint, we obtain a lower bound on the approximation-ratio of these algorithms. Since the function $c(a, b)$ has no critical points, its minimum must occur on the boundary of the constraint equation. Using Lagrange multipliers to minimize $a + b$ subject to $a + 4(1 + 4(a + b)) = ab$, we find $a = 16 + \sqrt{276}$ and $b = a + 1$, yielding a final approximation ratio of more than 1063.5.

## 4. Monotone Faraway Sampling

We turn now to the problem of producing a $k$-median candidate set in the insertion-only streaming setting. We note that this problem is related to, but distinct from, that discussed in the previous section. We begin by establishing relevant definitions to formalize our discussion of candidate sets in Section 1 and differentiate the notion of a candidate set from the related concept of a coreset. We then present the monotone faraway sampling (MFS) algorithm for producing a candidate set, and prove its correctness. Throughout this section, for ease of notation, we continue to suppress the metric space $\mathcal{X}$ from our notation, so that $\nu_{\mathrm{OPT}}(D, k)$ is understood to mean $\nu_{\mathrm{OPT}}(D, \mathcal{X}, k)$, where $\mathcal{X}$ is the set of all points appearing in stream $D$.

**Definition 13.** *Let $D$ be a stream of points from metric space $(\mathcal{X}, d)$ and let $P$ be the set of all points appearing in stream $D$. A set $F \subset P$ is a $(k, \lambda)$-*candidate set *for $P$ if there exists $C \subset F$ with $|C| = k$ satisfying $\nu(C, D) \leq \lambda \nu_{\text{OPT}}(D, k)$.*

Finding a candidate set for a stream $D$, in tandem with a coreset $S$ on $D$, allows one to find a clustering solution via a simple local search over the candidate set, similar to that in [16], using $S$ to (approximately) evaluate the cost of clustering with any $k$ centers from the candidate set. Our main result of this section shows that it is possible to construct a candidate set in the streaming setting in small space and with reasonable update time.

**Theorem 14.** *There exists an algorithm which, given a stream of points $D = p_1, p_2, \ldots p_n$ from metric space $(\mathcal{X}, d)$ and an integer $k \geq 1$, makes a single pass over the stream and produces a $(k, 1 + (1-\gamma)^{-1} + \rho)$-candidate set for $D$ with probability of failure $\eta$. The algorithm requires both space and update time of $O(k^2(\rho\gamma)^{-1} \log(k/\eta) \log(1 + k\gamma n))$.*

We claim that Algorithm 5 below is such an algorithm. Before presenting that algorithm and proving Theorem 14, we will first present a naïve solution to the candidate set problem, Algorithm 4. We will then alter this naïve algorithm to yield our Algorithm 5. We first establish a few definitions and supporting results.

**Definition 15.** *Given a set of points $P \subseteq \mathcal{X}$, we call a point $c^* \in P$ an optimal 1-median for $P$ if $\nu(P, \{c^*\}) = \nu_{\text{OPT}}(P, 1)$. Note that the optimal 1-median of a set need not be unique.*

**Definition 16.** *Given a (multi)set of points $P \subseteq \mathcal{X}$ and $0 \leq \gamma \leq 1$, we say a point $p \in P$ is $\gamma$-*good *for $P$ if there exists an optimal 1-median $c^* \in P$ such that there are at most $\gamma|P|$ points in $P$ that are closer to $c^*$ than $p$ is.*

When $\gamma = 1/2$, this definition coincides with the definition of "good points" given in [5]. As such, we have an analogue to Lemma 2.1 of [5].

**Lemma 17.** *If a point $p \in P$ is $\gamma$-good for $P$, then $\nu(P, \{p\}) \leq (1 + (1-\gamma)^{-1})\nu_{\text{OPT}}(P, 1)$.*

*Proof.* The proof is a simple extension of Lemma 2.1 in [5]. Let $p \in P$ be $\gamma$-good for $P$ and let $c^* \in P$ be the optimal 1-median guaranteed by Definition 16. Define $B = \{q \in P \mid d(p, c^*) \leq d(q, c^*)\}$. Summing over all $q \in B$ and using the fact that $|B| \geq (1-\gamma)|P|$, we have

$$(1-\gamma)|P| d(p, c^*) \leq \sum_{q \in B} d(q, c^*). \tag{4}$$

It follows that

$$\nu(P,\{p\}) = \sum_{q\in P} d(q,p) \le \sum_{q\in P} (d(q,c^*) + d(p,c^*)) = \nu_{\mathrm{OPT}}(P,1) + |P|d(p,c^*)$$
$$\le \nu_{\mathrm{OPT}}(P,1) + \nu_{\mathrm{OPT}}(P,1)\frac{1}{(1-\gamma)} \sum_{q\in B} d(q,c^*) \le \left(1 + \frac{1}{(1-\gamma)}\right)\nu_{\mathrm{OPT}}(P,1),$$

where the inequalities follow from the triangle inequality, Equation (4) and the fact that $\sum_{q\in B} d(q,c^*) \le \nu_{\mathrm{OPT}}(P,1)$, respectively. □

Lemma 17 suggests that we design a sampling algorithm that aims to capture one of the $1/\gamma$ proportion of points from each cluster that is $\gamma$-good for that cluster. This would require that we sample $\Omega(\gamma^{-1})$ proportion of the stream, and yet a larger proportion to ensure an exponentially small probability of failure. Note, however, that we would lose very little if instead of sampling a $\gamma$-good point, we managed to sample a point *close* to a $\gamma$-good point for a cluster. This idea motivates the following definition and counterpart to Lemma 17.

**Definition 18.** *For $\alpha \ge 0$ and $0 \le \gamma \le 1$, we say that a point $q \in \mathcal{X}$ is* $(\gamma,\alpha)$-decent *for $P$ if there exists a point $p \in P$ that is $\gamma$-good for $P$ and $d(q,p) \le \alpha/|P|$.*

**Lemma 19.** *If $p \in P$ is $(\gamma,\alpha)$-decent for $P$, then*

$$\nu(P,\{p\}) \le \left(1 + \frac{1}{1-\gamma}\right)\nu_{\mathrm{OPT}}(P,1) + \alpha.$$

*Proof.* The proof is analogous to that of Lemma 17, using a point $g$ that is $\gamma$-good for $P$ in place of $c^*$, followed by an application of Lemma 17 to bound $\nu(P,\{g\})$. □

Lemma 19 suggests a way to find a candidate set for a stream $D$. If $\gamma$ is a reasonably large constant, say, $\gamma = 1/2$, then for a cluster $C \subseteq D$ (i.e., a set of points all assigned to the same center under some optimal solution), there are by definition $\gamma|C|$ points in $C$ that are $\gamma$-good for $C$, and there are still more points within distance $\alpha/|C|$ of those $\gamma$-good points. Thus, intuitively speaking, a well-designed sampling scheme will, with high probability, sample one or more $(\gamma,\alpha)$-decent points for the set $C$. Algorithm 4 below maintains, in parallel, a number of correlated samples from the stream $D$. For a suitably-chosen set of nonnegative rationals $\mathcal{R}$, for each $r \in \mathcal{R}$, we will maintain a sample $F_r$. We call this parameter $r$ the *sample radius* of the sampler. Roughly speaking, this parameter controls how far away a new point must be from the existing sampled points for it to be considered a likely $(\gamma,\alpha)$-decent point. In the following subsections, we will show that we can choose this set $\mathcal{R}$ to be small, while still guaranteeing that with high probability, the union of these samples will contain a $(\gamma,\alpha)$-decent point for each cluster in the optimal solution on the entirety of the stream seen so far. Lemma 19 will then imply that this union includes a $(1 + (1-\gamma)^{-1} + \rho)$-approximate solution to the $k$-median problem on the stream.

**Algorithm 4** Algorithm for Monotone Faraway Sampling (MFS) from a stream of points. $\mathcal{R}$ is a set of non-negative rational numbers. We use $z \leftarrow \text{Uniform}(0,1)$ to mean that $z$ is assigned a value drawn uniformly at random from the interval $(0,1)$.

$N \leftarrow 1; \quad w \leftarrow \left(1 + \frac{4}{\rho\gamma}\right) k \log\left(\frac{k^2+k}{\eta}\right)$<br>
**for** $r \in \mathcal{R}$ **do**<br>
    $F_r^{(0)} \leftarrow \emptyset; \quad m_r^{(0)} \leftarrow 0$<br>
**end for**<br>
**while** $p_N \neq$ `END_OF_STREAM` **do**<br>
    $N \leftarrow N + 1; \quad z^{(N)} \leftarrow \text{Uniform}(0,1)$<br>
    **for** $r \in \mathcal{R}$ **do**<br>
        **if** $d(p_N, F_r^{(N-1)}) > r$ **then**<br>
            $m_r^{(N)} \leftarrow m_r^{(N-1)} + 1$<br>
            **if** $m_r^{(N)} z^{(N)} < w$ **then**<br>
                $F_r^{(N)} \leftarrow F_r^{(N-1)} \cup \{p_N\}$<br>
            **else**<br>
                $F_r^{(N)} \leftarrow F_r^{(N-1)}$<br>
            **end if**<br>
        **end if**<br>
    **end for**<br>
**end while**

Once we begin the iteration of the while-loop in Algorithm 4 (line 5) for point $p_N$ from stream $D$, we say that point $p_N$ has been *encountered*. The variable $N$ counts the number of points encountered. If $d(p_N, F_r^{(N-1)}) > r$ (line 8), we say that point $p_N$ is *considered* for sample $F_r$. If line 11 is executed, we say that point $p_N$ is *sampled* for sample $F_r$.

We will show that under the condition that we choose set $\mathcal{R}$ correctly, and provided that we can find a lower bound $V \leq \nu_{\mathrm{OPT}}(D, k)$ on the optimal cost, then when the above algorithm finishes, the set $F_V^* = \cup_{r\in\mathcal{R}} F_r$ will contain at least one $k$-element set that is a $(1 + (1-\gamma)^{-1} + \rho)$-approximation to the optimal $k$-median cost. In addition to proving this, we must do the following:

(1) Bound the number of values $r \in \mathcal{R}$ for which we must keep samples $F_r$.

(2) Show that with high probability, no sample $F_r$ grows larger than $O(w \log n)$, where $n$ is the length of the stream $D$, and $w$ is as defined on line 1.

### *4.1. Bounding the Number of Points Considered*

Let $D$ be a stream of points from a finite metric space. In what follows, let us fix positive integer $k$, $r \in \mathcal{R}$ and $\gamma \in (0, 1)$. Define $w$ as in line 1 of Algorithm 4. Let $A$ be a non-empty subset of points from stream $D$ and define the quantities

$$\ell(A) = \exp\left\{\frac{-w}{k + 2\nu_{\mathrm{OPT}}(D,k)/(r\gamma\,|A|)}\right\}, \quad \text{and} \quad h(A) = 2\nu_{\mathrm{OPT}}(D,k)/r + k\lceil\gamma\,|A|\rceil.$$

**Lemma 20.** *Given* $\lceil\gamma\,|A|\rceil$ *independent* $\mathrm{Uniform}(0,1)$ *random variables, the probability that each of them is at least* $w/h(A)$ *is no greater than* $\ell(A)$.

*Proof.* This follows immediately from the inequality $1 - x \leq \exp\{-x\}$ and the independence of the samples $z^{(N)}$ in Algorithm 4. □

In what follows, let $U_m$ denote the set of the first $m$ points considered for set $F_r$ in Algorithm 4.

**Lemma 21.** *For a point* $p$ *from stream* $D$*, let* $B_p$ *denote the set of points in stream* $D$ *within a distance of* $r/2$ *of* $p$*. For any point* $p$ *in stream* $D$*, with probability at least* $1 - \ell(A)$, $|B_p \cap U_{h(A)+1}| \leq \lceil\gamma\,|A|\rceil$.

*Proof.* Let $N_i$ be the values of $N$ for which $p_N \in B_p \cap U_{h(A)}$, sorted in increasing order so that $N_1 < N_2 < \cdots < N_M$. There will thus be a $N_i$ for each $1 \leq i \leq M := \left|B_p \cap U_{h(A)}\right|$.

By Lemma 20, the probability that each of the elements of $\{Z_i \mid 1 \leq i \leq \lceil\gamma\,|A|\rceil\}$ takes on a value of at least $w/h(A)$ is at most $\ell(A)$. Otherwise, $z^{(N_i)} < w/h(A)$ for at least one $i$ in $1 \leq i \leq \lceil\gamma\,|A|\rceil$. Since $p_{N_i} \in U_{h(A)}$, $p_{N_i}$ was one of the first $h(A)$ points to be considered and thus $m^{(N_i)} \leq h(A)$. It follows that $m^{(N_i)} z^{(N_i)} < w$, so $p_{N_i}$ is sampled. After this time, no more points in $B_p$ can be considered after $p_{N_i}$ is sampled for $F_r$, because $p_{N_i} \in B_p$ and $\operatorname{diam} B_p \leq r$ jointly imply that the condition $d(p_{N_i}, F_r) > r$ can never be satisfied for any future $p_{N_i} \in B_p$. Thus the only points in $B_p \cap U_{h(A)+1}$ can be $p_{N_1}, p_{N_2}, \ldots, p_{N_i}$. Since $i \le \lceil \gamma |A| \rceil$, there are at most $\lceil \gamma |A| \rceil$ such points. □

**Lemma 22.** *Let $\beta > 0$. If $C = \{c_1, c_2, \ldots, c_k\}$ is an optimal solution for the $k$-median problem on stream $D$, then $D$ contains at most $\beta^{-1}\nu_{\text{OPT}}(D,k)$ points at a distance of more than $\beta$ from $C$.*

*Proof.* Suppose there are more than $\nu_{\text{OPT}}(D,k)/\beta$ points $x$ with $d(x,C) > \beta$. Then the cost of clustering with $C$ is $\nu_{\text{OPT}}(D,k) = \sum_i d(p_i, C) > \beta \nu_{\text{OPT}}(D,k)/\beta = \nu_{\text{OPT}}(D,k)$, a contradiction. □

**Lemma 23.** *Let $A$ be a collection of points from stream $D$ and let $r > 0$ be some number included in set $\mathcal{R}$ in Algorithm 4. The probability that Algorithm 4 considers strictly more than $h(A)$ points for set $F_r$ is at most $k\ell(A)$.*

*Proof.* Let $C$ be an optimal solution for the $k$-median problem on $D$. Applying a union bound over the $k$ points in $C$, Lemma 21 implies that with probability at least $1 - k\ell(A)$, the set $U_{h(A)+1}$ contains at most $k\lceil \gamma |A| \rceil$ points within a distance of $r/2$ of set $C$, Condition on this event and suppose by way of contradiction that more than $h(A)$ points are considered for set $F_r$. By definition, the set of considered points $U_{h(A)+1}$ has cardinality $h(A) + 1 = 2\nu_{\text{OPT}}(D,k)/r + k\lceil \gamma |A| \rceil + 1$. By Lemma 22, there are at most $2\nu_{\text{OPT}}(D,k)/r$ points at a distance of more than $r/2$ of $C$. Thus, even if $U_{h(A)+1}$ contains all such points, $U_{h(A)+1}$ will still contain at most $2\nu_{\text{OPT}}(D,k)/r + k\lceil \gamma |A| \rceil$ points in total, contradicting the cardinality of $U_{h(A)+1}$. □

### *4.2. Finding a Decent Point*

We have just shown in Lemma 23 that for a given parameter $r \in \mathcal{R}$, at most a certain number of points from the stream are considered for the sample $F_r$. We turn now to the problem of showing that this sample contains a $(\gamma, \alpha)$-decent point for suitably-chosen $\alpha$. As before, fix integer $k$, $r \in \mathbb{Q}_{\ge 0}$ and $\gamma \in (0,1)$ and let $w$ be as in line 1 of Algorithm 4.

**Lemma 24.** *Let $A$ be a collection of points from stream $D$. In Algorithm 4, either*

*(i) every point $p_i$ that is* $\gamma$-good *for $A$ is considered for $F_r$, or*

*(ii) a point $p_i$ that is* $(\gamma, r|A|)$-decent *for $A$ is sampled for $F_r$.*

*Proof.* Let a point $p = p_N$ be a $\gamma$-good point for $A$, and let $F_r^{(N-1)}$ be the set of points that have been sampled for $F_r$ when point $p$ is encountered. At time $N$ when $p$ is encountered, $p$ is considered for $F_r$ if and only if $d(p, F_r^{(N-1)}) > r$. Therefore, if we do not consider $p$ for $F_r$, then $d(p, F_r^{(N-1)}) \le r$, implying the existence of a point $s_p \in F_r^{(N-1)}$ with $d(s_p, p) \le r$. Since $p$ is a $\gamma$-good point for $A$, by definition, $s_p$ must be $(\gamma, r|A|)$-decent for $A$. Thus, unless Algorithm 4 considers every $\gamma$-good point for $A$, it must sample a $(\gamma, r|A|)$-decent point for $A$. □

**Lemma 25.** *Let $A$ be a collection of points from stream $D$. On termination of Algorithm 4, sample $F_r$ contains a $(\gamma, r|A|)$-*decent *point for $A$ with probability at least* $1-(k+1)\ell(A)$.

*Proof.* Consider the scenario where Algorithm 4 does not include any $(\gamma, r|A|)$-decent points for $A$ in sample $F_r$. By Lemma 24, this means that Algorithm 4 considers all of the each of the at least $\lceil \gamma |A| \rceil$ points that are $\gamma$-good for $A$.

By applying Lemmas 23 and 20 followed by a union bound, we see that the probability that either

(i) Algorithm 4 considers more than $h(A)$ points for $F_r$, or

(ii) the values $z^{(N)}$ drawn when each of the $\gamma$-good points are considered for $F_r$ are each at least $w/h(A)$

is at most $(k+1)\ell(A)$.

Excluding this event, Algorithm 4 must consider at most $h(A)$ points for sample $F_r$, including each of the $\gamma$-good points for $A$, and at least one of those good points has a corresponding random value of $z^{(i)} < w/h(A)$. Let $p_i$ denote one of these points. Since the total number of points considered by Algorithm 4 is at most $h(A)$, it follows that $m^{(i)}$ is at most $h(A)$, and $m^{(i)}z^{(i)} < h(A)w/h(A) = w$, so $p_i$ will be sampled for $F_r$. Since $p_i$ is $\gamma$-good for $A$, it is trivially $(\gamma, r|A|)$-decent for $A$. Thus, with probability at least $1-(k+1)\ell(A)$, Algorithm 4 includes a $(\gamma, r|A|)$-decent point for $A$ in sample $F_r$. □

We are ready to establish one of the main results pertaining to Algorithm 4, showing that for suitable values of $r \in \mathcal{R}$, the algorithm includes decent points in $F_r$.

**Lemma 26.** *For any set of points $A$ from stream $D$, if*

$$w = \left(1 + \frac{4}{\rho\gamma}\right) k \log\left(\frac{k^2+k}{\eta}\right) \quad \textit{and} \quad r \in \left(\frac{\rho}{2k}\frac{\nu_{\mathrm{OPT}}(D,k)}{|A|}, \frac{\rho}{k}\frac{\nu_{\mathrm{OPT}}(D,k)}{|A|}\right],$$

*then with probability at least $1-\eta/k$, Algorithm 4 will include in sample $F_r$ at least one* $(\gamma, \rho\nu_{\mathrm{OPT}}(D,k)/k)$-decent *point for $A$.*

*Proof.* By choice of $r$ and $w$, we have

$$\exp\left\{\frac{-w}{2\nu_{\mathrm{OPT}}(D,k)/(r\gamma|A|)+k}\right\} \le \exp\left\{\frac{-w}{4k/(\rho\gamma)+k}\right\}$$
$$= \exp\left\{-\frac{\left(1+\frac{4}{\rho\gamma}\right)k\log\left(\frac{k^2+k}{\eta}\right)}{4k/(\rho\gamma)+k}\right\} = \exp\left\{-\log\left(\frac{k^2+k}{\eta}\right)\right\} = \frac{\eta}{k^2+k}.$$

Furthermore, since $r \le \rho\nu_{\mathrm{OPT}}(D,k)/(k|A|)$, any $(\gamma, r|A|)$-decent point for $A$ is also $(\gamma, \rho\nu_{\mathrm{OPT}}(D,k)/k)$-decent for $A$. Therefore, by Lemma 25, Algorithm 4 includes in sample $F_r$ a $(\gamma, \rho\nu_{\mathrm{OPT}}(D,k)/k)$-decent point for $A$ with probability at least $1-(k+1)\frac{\eta}{k^2+k} = 1-\eta/k$. □

In the sequel, we will set $A$ to be an optimal cluster of $D$, and use the above result to obtain $(\gamma, \rho\nu_{\mathrm{OPT}}(D,k)/k)$-decent points for each cluster, from which we will construct a $1+(1-\gamma)^{-1}+\rho$ approximation for the $k$-median problem by selecting values of $r$ appropriately.

### *4.3. Exponentially Spaced Estimates for r*

Our result in the previous subsection depended on being able to select an appropriate value for $r$. However, since both $\nu_{\text{OPT}}(D, k)$ and $|A|$ are unknown a priori, we do not know which value to use for $r$, and Algorithm 4 as it stands is insufficient. In this and the following sections, we will address this issue by demonstrating a scheme whereby we only need to store the values for $O(k)$ "guesses" for $r$ in such a way that we will be guaranteed to include in the set $\mathcal{R}$ the values that ensure that Lemma 26 applies for each cluster in the optimal solution.

In what follows, we fix $\rho \in (0, 1)$. For $V > 0$, let $T_V = \{2^i \mid i \in \mathbb{Z}, 2^i > \rho V/(2kn)\}$ and for $a \geq \rho V/(2kn)$, we define

$$T_V(a) = \begin{cases} 0 & \text{if } a \leq \frac{\rho V}{2kn} \\ 2^{\lfloor \log a \rfloor} & \text{otherwise.} \end{cases}$$

**Observation 27.** *Let $a \geq \rho V/(kn)$. With notation as above,*

*(a)* $T_V(a) \in (a/2, a]$

*(b)* $T_V(a) \in T_V$.

**Lemma 28.** *Let $A$ be a collection of points from stream $D$. If*

$$w = \left(1 + \frac{4}{\rho\gamma}\right) k \log\left(\frac{k^2 + k}{\eta}\right) \quad \textit{and} \quad r = T_V\left(\frac{\rho}{k\,|A|}\nu_{\text{OPT}}(D, k)\right) \in \mathcal{R},$$

*then with probability at least $1 - \eta/k$, upon termination of Algorithm 4, sample $F_r$ will include at least one* $(\gamma, \rho\nu_{\text{OPT}}(D, k)/k)$-decent *point for $A$.*

*Proof.* Since $T_V(a) \in (a/2, a]$, this follows from Lemma 26. □

**Corollary 29.** *Let $A$ be a collection of points from stream $D$. If $V \leq \nu_{\text{OPT}}(D, k)$ and $r \in T_V$, then with probability at least $1 - \eta/k$ Algorithm 4 will include at least one* $(\gamma, \rho\nu_{\text{OPT}}(D, k)/k)$-decent *point for $A$ in sample $F_r$.*

*Proof.* We have $\rho\nu_{\text{OPT}}(D, k)/(k\,|A|) \geq \rho V/kn$, so by Observation 27, $T_V(\rho\nu_{\text{OPT}}(D, k)/(k\,|A|)) \in T_V$. Therefore, $r = T_V(\rho\nu_{\text{OPT}}(D, k)/(k\,|A|)) \in T_V$. Accordingly, Lemma 28 implies that for this choice of $r$, Algorithm 4 will include in sample $F_r$ at least one $(\gamma, \rho\nu_{\text{OPT}}(D, k)/k)$-decent point for $A$ with probability at least $1 - \eta/k$. □

We are now ready to state our second major theorem, which shows that we can choose $\mathcal{R}$ in such a way that Algorithm 4 will yield a candidate set for the stream. Specifically, Theorem 30 shows that by taking $\mathcal{R} = T_V$ in Algorithm 4, provided $V$ is chosen correctly, we will find a decent point for each of the clusters in the optimal solution with high probability, implying that the union of the samples maintained by Algorithm 4 is a candidate set for the stream.

**Theorem 30.** *Let $F_V^* = \cup_{r \in T_V} F_r$, that is, choose $\mathcal{R}$ in Algorithm 4 to be $T_V$. If $V \leq \nu_{\mathrm{OPT}}(D,k)$, then with probability at least $1-\eta$, $F_V^*$ contains at least one $k$-element subset $C$ such that $\nu(D,C) \leq \big(1+(1-\gamma)^{-1}+\rho\big)\nu_{\mathrm{OPT}}(D,k)$.*

*Proof.* Let $\{A_1, A_2, \cdots, A_k\}$ be an optimal $k$-clustering of $D$. By Corollary 29 and a union bound over the $k$ clusters, it follows that with probability at least $1-\eta$, on termination of Algorithm 4, $F_V^*$ contains a $(\gamma, \rho\nu_{\mathrm{OPT}}(D,k)/k)$-decent point for each of the clusters. Choose a set of $k$ centers $C$ by choosing from each $A_i$ some such decent point $c_i \in A_i$. The $c_i$ need not be distinct— if they are not, we simply assume that set $C$ has been padded with arbitrary points from $F_V^*$ to ensure that $|C| = k$. Then,

$$\begin{aligned}\nu(D,C) &\leq \sum_{i=1}^{k}\sum_{x \in A_i} d(x,c_i) = \sum_{i=1}^{k}\nu(A_i,\{c_i\}) \\ &\leq \rho\nu_{\mathrm{OPT}}(D,k) + \sum_{i=1}^{k}\left(1+\frac{1}{1-\gamma}\right)\nu_{\mathrm{OPT}}(A_i,1) = \left(1+\frac{1}{1-\gamma}+\rho\right)\nu_{\mathrm{OPT}}(D,k),\end{aligned}$$

as we wished to show. □

### *4.4. Using ranges of r values*

As it stands, Theorem 30 is still not of much use to us. Even if we did have a lower bound $V$ for $\nu_{\mathrm{OPT}}(D,k)$, $T_V$ would still be an infinite set, meaning that in Algorithm 4, we would have to maintain samples $F_r$ for an infinite number of values of $r$. In this section, we address this issue by showing that

(1) We need not actually keep an MFS sample for every possible value of $r$ in $T_V$. Instead, we can group values of $r$ into ranges of the form $[\ell, u]$ so that $F_{r_1} = F_{r_2}$ for all $r_1, r_2 \in [\ell, u]$.

(2) We can bound the number of such ranges for which we must maintain samples.

(3) We can indeed find a lower bound $V \leq \nu_{\mathrm{OPT}}(D,k)$ so that Theorem 30 applies.

Algorithm 5 alters Algorithm 4 to take these points and our above results into account, and its correctness, claimed in Theorem 14, will follow immediately once we address the above three points, handled in Lemmas 31, 33 and 34, respectively.

We address Item 1 first by noting that for $r_1, r_2 \in \mathcal{R}$ in Algorithm 4 with $r_1 < r_2$, $F_{r_1}^{(N)} = F_{r_2}^{(N)}$ for all time steps $N$ until we encounter a point $x$ in the stream whose distance from $F_{r_1}$ is between $r_1$ and $r_2$.

**Lemma 31.** *Let $0 \leq r_1 < r_2$ be two values from $\mathcal{R}$ in Algorithm 4 and let $N$ be the time of arrival of some point in the stream. If there does not exist $i \in [N]$ for which $r_1 < d(p_i, F_{r_1}^{(i)}) \leq r_2$, then $F_{r_1}^{(N)} = F_{r_2}^{(N)}$.*

*Proof.* We proceed by induction on time step $N$. Note that before encountering the $N$-th element of the stream $p_N$, the state of out algorithm is captured entirely by the contents of the sets $F_r^{(N)}$ for all $r \in \mathcal{R}$. At the start of the algorithm, $F_r^{(0)} = \emptyset$ for all possible values $r \in \mathcal{R}$ by definition, establishing the base case $N = 0$.

Consider the inductive case where for all $N' < N$, we have $d(p_{N'}, F_{r_1}^{(N')}) \notin (r_1, r_2]$ and $d(p_{N'}, F_{r_2}^{(N')}) \notin (r_1, r_2]$. By our induction hypothesis, we have $F_{r_1}^{(N-1)} = F_{r_2}^{(N-1)}$. Thus, if $d(p_N, F_{r_1}^{(N')}) \notin (r_1, r_2]$ and $d(p_N, F_{r_2}^{(N')}) \notin (r_1, r_2]$, then either $p_N$ will be considered for both sets $F_{r_1}^{(N)}$ and $F_{r_2}^{(N)}$ or it will be considered for neither. Observe that $m_{r_1}^{(N)} = m_{r_2}^{(N)}$ by similar reasoning as that used to show that the sets are identical at time $N-1$. Therefore, if $p_N$ is considered, then since the inclusion or exclusion of point $p_N$ depends only on the outcome of the random draw $z^{(N)}$ and on $m_{r_1}^{(N)}$, point $p_N$ is included in sample $F_{r_1}^{(N)}$ if and only if it is included in $F_{r_2}^{(N)}$, and thus $F_{r_1}^{(N)} = F_{r_2}^{(N)}$. $\square$

Lemma 31 suggests a scheme for keeping track of $F_r$ over a variety of values for $r$. Rather than maintaining $F_r$ for all $r \in \mathcal{R}$, we keep track of $F_r$ for all $r$ in a given *range* of values on which $F_r$ has the same state. Let us use $\mathcal{T} = \{R_1, R_2, \dots\}$ to denote a collection of disjoint intervals, which we will call *ranges*. Rather than keeping sample $F_r$ for every distinct value of $r$ from $\mathcal{R}$, we will maintain one sample for each range and maintain the invariant that this sample is equivalent to $F_r$ for any $r$ in its corresponding range. That is, for each $R_i \in \mathcal{T}$, we will maintain a sample $F_{R_i}$ so that $F_r = F_{R_i}$ for any $r \in R_i$.

Initially, we need only concern ourselves with a single range, $[0, \infty)$, since before we've seen any points from the stream, $F_r = \emptyset$ for any sample radius $r \in \mathcal{R}$. Whenever a point $p_N$ is encountered, we examine its distance to each of the samples $F_{R_i}$ for each $R_i \in \mathcal{T}$. If the distance of point $p_N$ to a sample falls within the range of that sample, for example, if $d(p_N, F_{R_i}) \in R_i = [\ell, u]$, then we split range $R_i$ into two and "trim" the ends of these new ranges to discard values of $r$ that do not concern us. Specifically, we split $R_i$ into two intervals, one of values from $R_i$ that are less than $d(p_N, F_{R_i})$ and one of values from $R_i$ that are greater than $d(p_N, F_{R_i})$. Then, we discard parts of ranges that do not include any elements from set $T_V$ as defined in Subsection 4.3.

A few loose ends still remain. Firstly, we must show that we can avoid tracking too many different ranges at once. Neither Algorithm 4 nor Algorithm 5 will suffice in the streaming domain if we must maintain, say, $\Theta(n)$ different samples to cover all the ranges in $\mathcal{T}$. Secondly, Theorem 30 yields a crucial property of our algorithm, but it rests on our having a lower bound $V$ for $\nu_{\mathrm{OPT}}(D, k)$. Finally, some of our preceding reasoning has assumed that $n$, the number of elements in the stream, is known ahead of time, an unrealistic assumption under the streaming model. We will show how this assumption can be relaxed.

**Definition 32.** *Let $R = [\ell, u)$ be an interval with $0 \le \ell < u$, let $S$ be a set of points from metric space $(\mathcal{X}, d)$ and let $m$ be a non-negative integer. A* ranged monotone faraway sample *(RMFS) is a triple $(R, S, m)$, where $R = [\ell, u)$, $S$ is a set of sampled points and $m$ is the number of points considered for sample $S$.*

Note that using the notation from Algorithm 4, if $(R, S, m)$ is a RMFS, then $r \in R$ implies $S = F_r$.

**Lemma 33.** *If $U > \nu_{\mathrm{OPT}}(D,k)$, then there exists an index set $\mathcal{I} \subset \mathbb{Z}$ with $|\mathcal{I}| \leq 3(k-1)$ such that for all pairs of points $p, q$ from stream $D$, either $d(p,q) < 2U$ or $d(p,q) \in \cup_{i\in\mathcal{I}}[2^i, 2^{i+1}]$.*

*Proof.* Let $D = p_1, p_2, \ldots, p_n$ be a stream of points from a metric space and assume $U > \nu_{\mathrm{OPT}}(D,k)$. Let $A_1, A_2, \ldots, A_k$ be an optimal $k$-clustering of the stream with cost $\nu_{\mathrm{OPT}}(D,k)$. We associate to the stream $D$ a complete, undirected, weighted graph $G$, with a node for each point $p$ from the stream and an edge $(p_i, p_j)$ for each $i \neq j$ with $i, j \in [n]$ Each such edge has non-negative edge weight $w(p_i, p_j) = d(p_i, p_j)$. Let $L$ denote the set of all such edge weights in graph $G$.

Call edge $(u, v)$ *augmenting* for weighted undirected graph $H$ if $u$ and $v$ are disconnected in $H$. Call $\delta$ an *intra-component* distance for graph $H$ if there exists a pair of nodes $u$ and $v$ in the same connected component of $H$ for which $d_H(u,v) = \delta$, where $d_H$ denotes the shortest path distance in graph $H$. Now, consider the graph $G_0 = T_1^{(0)} \cup T_2^{(0)} \cup \cdots \cup T_k^{(0)}$, where each of the $T_i^{(0)}$ is the (complete) subgraph in $G$ induced by optimal cluster $A_i$. By construction, $G_0$ includes all the points in stream $D$. Further, since $A_1, A_2, \ldots, A_k$ is a partition of the stream, $G_0$ consists of exactly $k$ connected components. By our assumption that $A_1, A_2, \ldots, A_k$ is an optimal clustering, we know that for all $i$, $\mathrm{diam}(A_i) \leq 2\nu_{\mathrm{OPT}}(D,k) < 2U$.

Starting with graph $G_0$, choose the shortest augmenting edge (call it $e_1$) for $G_0$ and add it to $G_0$. Let $T_i^{(0)}$ and $T_j^{(0)}$ be the two connected components in $G_0$ joined by edge $e_1$. For all nodes $u$ in component $T_i^{(0)}$ and all nodes $v$ in component $T_j^{(0)}$, add edge $(u,v)$ to $G_0$ with weight $d(u,v)$. Call the resulting graph $G_1$. $G_1$ consists of $k-1$ connected components.

Repeat this operation, adding the shortest remaining augmenting edge $e_2$ for $G_1$ and adding all edges between pairs of newly-connected nodes to obtain graph $G_2$, which consists of $k-2$ connected components. Performing this operation $k-1$ times in total, we obtain a connected graph $G_{k-1}$. For $i = 1, 2, \ldots, k-1$, let $a_i$ be the length of edge $e_i$ (i.e., the shortest augmenting edge for graph $G_{i-1}$), let $b_i$ be the diameter of the component in graph $G_i$ that was created by merging two components from graph $G_{i-1}$, let $c_i = \max_{1\leq j\leq i} b_j$, and let $L_i = L_{i-1} \cup [a_i, c_i]$. Set $L_0 = [0, 2U)$ and $c_0 = \max_{1\leq i\leq k} \mathrm{diam}(T_i^{(0)})$. Since $G_0$ has diameter at most $2U$, $L_0$ includes all intra-component distances in $G_0$. Observe that the set of intra-component distances in graph $G_i$ that are not intra-component distances in graph $G_{i-1}$ all lie in the range $[a_i, c_i]$, since (1) $a_i$ is a lower bound on the distance between two nodes that are not in the same connected component in graph $G_{i-1}$ and (2) $\mathrm{diam}(G_i) = b_i \leq c_i$ by definition of $c_i$.

By construction, $a_i \geq a_{i-1}$ for all $1 < i \leq k-1$, and since the diameter of the new component formed when we add edge $e_i$ to graph $G_{i-1}$ is precisely the length of edge $e_i$ plus the diameters of the merged components, we have $b_i \leq a_i + 2c_{i-1}$ (where we have used the fact that $c_{i-1}$ is an upper bound on the diameter of any component in graph $G_{i-1}$). Further, we have that $c_i = \max\{c_{i-1}, b_i\} = \max\{c_{i-1}, a_i + 2c_{i-1}\} = a_i + 2c_{i-1}$. Thus, for all $i \in [k-1]$, we have either

(i) $c_{i-1} \leq a_i$, in which case $[a_i, c_i] \subset [a_i, a_i + 2c_{i-1}] \subset [a_i, 3a_i]$, or

(ii) $c_{i-1} > a_i$, in which case

$$[a_i, c_i] \subset [a_i, a_i + 2c_{i-1}] \subset [a_{i-1}, 3c_{i-1}] = [a_{i-1}, c_{i-1}] \cup [c_{i-1}, 3c_{i-1}] \subset L_{i-1} \cup [c_{i-1}, 3c_{i-1}].$$

It follows that $L_i \setminus L_{i-1}$ is contained in either $[a_i, 3a_i]$ or $[c_{i-1}, 3c_{i-1}]$. In either case, $L_i \setminus L_{i-1}$ intersects at most 3 distinct ranges of the form $[2^j, 2^{j+1}]$ for some integer $j$. Thus, $L_{k-1} \setminus L_0 = L_{k-1} \setminus [0, 2U) \supset L \setminus [0, 2V)$ can intersect at most $3(k-1)$ distinct ranges of the form $[2^j, 2^{j+1}]$. □

**Lemma 34.** *Let $\mathcal{L}^{(N)}$ be the set of RMFS objects maintained by Algorithm 5 at some time $N$ upon the conclusion of an iteration of the while-loop on line 2. Let $R_1 = [\ell_1, \infty), R_2 = [\ell_2, u_2), \ldots, R_a = [\ell_a, u_a)$ be the ranges of the RMFS objects contained in $\mathcal{L}^{(N)}$, with $\ell_1 > \ell_2 > \cdots > \ell_a$. If $a \geq 3k - 1$, then $\ell_a \leq \nu_{\mathrm{OPT}}(D, k)$.*

*Proof.* By construction of the algorithm, ranges $R_1, R_2, \ldots, R_a$ are all disjoint, and for all $i \in [a]$, $\ell_i = 2^{j_i}$ for some $j_i \in \mathbb{Z}$. Let $B = [\ell_a, \infty) \setminus (\cup_{i=1}^a R_i)$. For each $i = 1, 2, \ldots, a-1$, the interval $[u_i, \ell_{i+1})$ contains an interval of the form $[2^j, 2^{j+1})$ for some $j \in \mathbb{Z}$, and for each such $i$, by construction of our algorithm, there must have been in stream $D$ a pair of points $u, v$ with $d(u, v) \in [u_i, \ell_{i+1})$. Since $[u_i, \ell_{i+1}]$ is the union of one or more intervals of the form $[2^j, 2^{j+1})$, $d(u, v)$ is contained in an interval of the form $[2^j, 2^{j+1}]$. Such a pair of points exists for each $i \in [a-1]$, and each such pair of points corresponds to a distinct interval of the form $[2^j, 2^{j+1}]$. Thus, if $a \geq 3(k-1) + 2$, the converse of Lemma 33 implies that $\ell_a/2 \leq \nu_{\mathrm{OPT}}(D, k)$, since we have shown that pairs of points from stream $D$ lie in strictly more than $3(k-1)$ intervals of the form $[2^j, 2^{j+1}]$. □

With the above results in hand, we are ready to prove Theorem 14. Recall that the result states that given a data stream $D$, Algorithm 5 produces, with probability at least $1-\eta$, a $(k, 1+(1-\gamma)^{-1}+\rho)$-candidate set for $D$, and requires $O(k^2(\rho\gamma)^{-1} \log(k/\eta) \log(1 + k\gamma n))$ space and update time.

*Proof of Theorem 14.* Theorem 30 implies that with high probability, upon termination, the union of the samplers in Algorithm 14 is a $(k, 1 + (1-\gamma)^{-1} + \rho)$-candidate set for the stream provided that Algorithm 5 maintains the condition that the set $T_V$ is included in the union of ranges. This condition holds by construction of the algorithm, and the correctness of Algorithm 5 follows.

Lemma 23 shows that for any collection of points $A$, Algorithm 5 considers at most $h(A) = 2\nu_{\mathrm{OPT}}(D, k)/r + k\lceil \gamma\,|A| \rceil$ points for inclusion in a sample of radius $r$. The randomness introduced by the variables $z^{(N)}$ imply that we sample $wH_{h(A)}$ points in expectation, where $H_i$ denotes the $i$-th harmonic number. By construction, upon termination, any sample radius $r$ tracked by Algorithm 5 is within a multiplicative factor $O(\log k)$ of $\nu_{\mathrm{OPT}}(D, k)$, whence the trivial upper bound $|A| \leq n$ implies $H_{h(A)} = O(\log h(A)) = O(\log k(1 + \gamma n))$.

Standard concentration inequalities imply that with high probability, every sample $F_i$ maintained by Algorithm 5 contains $O(w \log h(A))$ points with high probability. Multiplying this by the $3k - 1$ RMFS

**Algorithm 5** Monotone Faraway Sampling using ranges for $r$. $k, \rho, \gamma$ and $\eta$ are user-supplied parameters. We assume that RMFS triples $(R_i, F_i, m_i)$, with $R_i, = [\ell_i, u_i)$, are stored in $\mathcal{L}$, indexed in descending order of their ranges' lower bounds. We use $\mathcal{L}[1..j]$ to denote the $j$ RMFS objects in $\mathcal{L}$ with the largest lower bounds.

$w \leftarrow \left(1 + \frac{4}{\rho\gamma}\right) k \log\left(\frac{k^2+k}{\eta}\right)$; $R_0 = [0, \infty)$; $m_0^{(0)} \leftarrow 0$; $\mathcal{L}^{(0)} \leftarrow \{(R_0, \emptyset, m_0^{(0)})\}$; $V \leftarrow 0$; $N \leftarrow 1$
**while** $p_N \neq$ `END_OF_STREAM` **do**
  $N \leftarrow N + 1$; $\mathcal{L}^{(N)} \leftarrow \emptyset$; $z^{(N)} \leftarrow \text{Uniform}(0, 1)$
  **for** $(R_i^{(N-1)}, F_i^{(N-1)}, m_i^{(N-1)}) \in \mathcal{L}^{(N-1)}$ **do**
    $\delta \leftarrow d(p_N, F_i^{(N-1)})$
    **if** $\delta < \ell_i$ **then**
      $\mathcal{L}^{(N)} \leftarrow \mathcal{L}^{(N)} \cup \{(R_i^{(N-1)}, F_i^{(N-1)}, m_i^{(N-1)})$
    **else if** $\delta \geq u_i$ **then**
      $m_i^{(N)} \leftarrow m_r^{(N-1)} + 1$
      **if** $m_i^{(N)} z^{(N)} < w$ **then**
        $F_i^{(N)} \leftarrow F_i^{(N-1)} \cup \{p_N\}$
      **else**
        $F_i^{(N)} \leftarrow F_i^{(N-1)}$
      **end if**
      $\mathcal{L}^{(N)} \leftarrow \mathcal{L}^{(N)} \cup \{(R_i^{(N-1)}, F_i^{(N)}, m_i^{(N)})\}$
    **else**
      $u' \leftarrow 2^{\lfloor \log \delta \rfloor}$; $\ell' \leftarrow 2^{\lceil \log \delta \rceil}$
      **if** $\ell_i \neq u'$ **then**
        $m_i^{(N)} \leftarrow m_i^{(N-1)} + 1$
        **if** $m_i^{(N)} z^{(N)} < w$ **then**
          $F_i^{(N)} \leftarrow F_i^{(N-1)} \cup \{p_N\}$
        **else**
          $F_i^{(N)} \leftarrow F_i^{(N-1)}$
        **end if**
        $R \leftarrow [\ell_i, u')$; $\mathcal{L}^{(N)} \leftarrow \mathcal{L}^{(N)} \cup \{(R, F_i^{(N)}, m_i^{(N)})\}$
      **end if**
      **if** $\ell' \neq u_i$ **then**
        $R \leftarrow [\ell', u_i)$; $\mathcal{L}^{(N)} \leftarrow \mathcal{L}^{(N)} \cup \{(R, F_i^{(N-1)}, m_i^{(N-1)})\}$
      **end if**
    **end if**
  **end for**
  $V \leftarrow \frac{1}{2}\ell_{3k-1}$; $i \leftarrow 3k - 1$; $\mathcal{L}^{(N)} \leftarrow \mathcal{L}^{(N)}[1..i]$
**end while**

objects maintained by the algorithm, we see that Algorithm 5 requires $O(k^2(\rho\gamma)^{-1}\log(k/\eta)\log k(1+\gamma n))$ space, and the same update time. □

## References

[1] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, K. Trinh, An improved approximation for k-median, and positive correlation in budgeted optimization, in: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '15, SIAM, 2015, pp. 737–756.
URL `http://dl.acm.org/citation.cfm?id=2722129.2722179`

[2] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: Theory and practice, IEEE Trans. on Knowl. and Data Eng. 15 (3) (2003) 515–528. `doi:10.1109/TKDE.2003.1198387`.
URL `http://dx.doi.org/10.1109/TKDE.2003.1198387`

[3] M. Charikar, L. O'Callaghan, R. Panigrahy, Better streaming algorithms for clustering problems, in: Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing, STOC '03, ACM, New York, NY, USA, 2003, pp. 30–39. `doi:10.1145/780542.780548`.
URL `http://doi.acm.org/10.1145/780542.780548`

[4] V. Braverman, A. Meyerson, R. Ostrovsky, A. Roytman, M. Shindler, B. Tagiku, Streaming k-means on well-clusterable data, in: Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, SIAM, 2011, pp. 26–40.
URL `http://dl.acm.org/citation.cfm?id=2133036.2133039`

[5] A. Meyerson, Online facility location, in: Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science, FOCS '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 426–.
URL `http://dl.acm.org/citation.cfm?id=874063.875567`

[6] M. Bādoiu, S. Har-Peled, P. Indyk, Approximate clustering via core-sets, in: Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing, STOC '02, ACM, New York, NY, USA, 2002, pp. 250–257. `doi:10.1145/509907.509947`.
URL `http://doi.acm.org/10.1145/509907.509947`

[7] K. Chen, On coresets for $k$-median and $k$-means clustering in metric and euclidean spaces and their applications, SIAM J. Comput. 39 (3) (2009) 923–947. `doi:10.1137/070699007`.
URL `http://dx.doi.org/10.1137/070699007`

[8] D. Feldman, M. Langberg, A unified framework for approximating and clustering data, in: Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing, STOC '11, ACM, New York, NY, USA, 2011, pp. 569–578. doi:10.1145/1993636.1993712.
URL http://doi.acm.org/10.1145/1993636.1993712

[9] S. Har-Peled, S. Mazumdar, Coresets for $k$-means and $k$-median clustering and their applications, in: STOC 2004, 2004, pp. 291–300.

[10] S. Har-Peled, A. Kushal, Smaller coresets for k-median and k-means clustering, Discrete Comput. Geom. 37 (1) (2007) 3–19. doi:10.1007/s00454-006-1271-x.
URL http://dx.doi.org/10.1007/s00454-006-1271-x

[11] J. L. Bentley, J. B. Saxe, Decomposable searching problems i. static-to-dynamic transformation, Journal of Algorithms 1 (4) (1980) 301 – 358. doi:http://dx.doi.org/10.1016/0196-6774(80)90015-2.
URL http://www.sciencedirect.com/science/article/pii/0196677480900152

[12] M. Bury, C. Schwiegelshohn, Random projections for k-means: Maintaining coresets beyond merge & reduce, CoRR abs/1504.01584.
URL http://arxiv.org/abs/1504.01584

[13] H. Fichtenberger, M. Gillé, M. Schmidt, C. Schwiegelshohn, C. Sohler, BICO: BIRCH meets coresets for k-means clustering, in: Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings, 2013, pp. 481–492. doi:10.1007/978-3-642-40450-4_41.
URL http://dx.doi.org/10.1007/978-3-642-40450-4_41

[14] M. Shindler, A. Wong, A. W. Meyerson, Fast and accurate k-means for large datasets, in: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 24, Curran Associates, Inc., 2011, pp. 2375–2383.
URL http://papers.nips.cc/paper/4362-fast-and-accurate-k-means-for-large-datasets.pdf

[15] S. Guha, Tight results for clustering and summarizing data streams, in: Proceedings of the 12th International Conference on Database Theory, ICDT '09, ACM, New York, NY, USA, 2009, pp. 268–275. doi:10.1145/1514894.1514926.
URL http://doi.acm.org/10.1145/1514894.1514926

[16] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit, Local search heuristic for $k$-median and facility location problems, SIAM Journal on Computing 33 (3) (2004) 544–562.