Accurate Low-Space Approximation of Metric k-Median for Insertion-Only Streams

Vladimir Braverman[1], Harry Lang[2], and Keith Levin[1](✉)

[1] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
klevin@jhu.edu

[2] Department of Mathematics, Johns Hopkins University, Baltimore, MD, USA

Abstract. We present a low-constant approximation for metric k-median on an insertion-only stream of n points 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. It is well-known that running an offline algorithm on this bicriterion solution yields a $(17.66+\epsilon)$-approximation.

Previously, there have been two lines of research that trade off between space and accuracy in the streaming k-median problem. To date, the best-known (k,ϵ)-coreset construction requires $O(\epsilon^{-2}k\log^4 n)$ space [8], while the best-known $O(k\log n)$-space algorithm provides only a $(O(k\log n), 1063)$-bicriterion [3]. Our work narrows this gap significantly, matching the best-known space while significantly improving the accuracy from 1063 to $2+\epsilon$. We also provide a matching lower bound, showing that any polylog(n)-space streaming algorithm that maintains an (α,β)-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 a storing an accurate summary of recent segments and a lower-space summary of older segments, our algorithm maintains a $(O(\epsilon^{-3}k\log n), 2+\epsilon)$-bicriterion solution for the entire input.

Keywords: Streaming algorithms · k-median · Clustering

1 Introduction

In metric k-median clustering over insertion-only streams, we are sequentially given n points from a metric space and attempt to return a set of k centers that

V. Braverman—This material is based upon work supported in part by the National Science Foundation under Grants IIS-1447639 and CCF-1650041.

H. Lang—This research is 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.

D. Gaur and N.S. Narayanaswamy (Eds.): CALDAM 2017, LNCS 10156, pp. 72–82, 2017.
DOI: 10.1007/978-3-319-53007-9_7

approximately minimize the sum of the distances of each point to its nearest center. We present an improved algorithm for this problem, maintaining the best-known space bound while drastically improving the approximation-ratio.

Streaming clustering has a long history since the work of Guha, Meyerson, Mishra, Motwani and O'Callaghan [11]. There have been two main classes of algorithms that have polylogarithmic space complexity and solve the streaming version of metric k-median. The first class contains facility-based algorithms, starting with the first polylogarithmic solution for the streaming k-median by Charikar, O'Callaghan, and Panigrahy [6]. These methods build upon the connection between the k-median problem and the facility location problem, using the online algorithm of Meyerson [14] as a subroutine. Facility-based algorithms achieve low storage (currently $O(k \log n)$-space due to [3]), but suffer from an extremely large approximation ratio. The best-space algorithm in this class provides a $(O(k \log n), 1063)$-bicriterion (see Sect. 7.2 for a calculation of this constant). The second class contains coreset-based algorithms, such as the works of [1,7,8,12,13]. By coreset-based, we refer to algorithms that can return a (k, ϵ)-coreset for any $\epsilon > 0$. A (k, ϵ)-coreset of a set A is a set B such that the cost of clustering A and B with any set of centers differ by at most a factor of $(1 \pm \epsilon)$. These algorithms achieve an arbitrarily low approximation ratio, but yet require significantly more storage, the lowest being a $O(\epsilon^{-2} k \log^4 n)$-space coreset due to [8]. In this line of research, an offline coreset construction is provided, which is then transformed into a streaming construction using the merge-and-reduce technique of [2] from 1980. Merge-and-reduce multiplies the space-bound by a factor of $\Omega(\log^3 n)$, and although other methods have been found for the Euclidean case [4,9], this remains the only technique available for coresets in general metric spaces. Without overcoming this 35-year-old barrier, coreset-based algorithms cannot match the space-bounds of facility-based algorithms.

The two classes of algorithms suggest a possible trade-off between the favorable space-bounds of facility-based methods and the favorable approximation ratio of coreset-based methods. A natural question is if 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 [15]. We now answer this in the affirmative for general metric spaces, using a technique entirely different from that of [15]. Our algorithm achieves a low-approximation ratio using $O(\epsilon^{-3} k \log n)$-space and maintaining a $(O(\epsilon^{-3} k \log n), 2 + \epsilon)$-bicriterion.

In 2009, an important result by Guha [10] was a facility-based $(34 + \epsilon)$-approximation using $O(\epsilon^{-3} \log \frac{1}{\epsilon} k \log^2 n)$-space. This straddles the above-mentioned space-accuracy trade-off by offering a low-constant (although not as low as offered by coreset-based algorithms) as well as low-space (although not as low as the $O(k \log n)$ offered by facility-based algorithms). In comparison, our algorithm offers both lower space and a lower approximation ratio than [10]. It is a well-known result [3,6,10,11] that running an offline γ-approximation on a (α, β)-bicriterion solution yields a $(\beta + 2\gamma(1 + \beta))$-approximation. With our $(O(\epsilon^{-3} k \log n), 2 + \epsilon)$-bicriterion, running the offline 2.61-approximation of [5] yields a $(17.66 + \epsilon)$-approximation. In relation to [10], this is a 48% reduction in the approximation factor and the space requirement is improved from $O(k \log^2 n)$ to $O(k \log n)$. Additionally, we show in the Appendix (see Sect. 7.1) that no polylog(n)-space algorithm can improve upon our approximation ratio.

2 Our Contribution

We present an algorithm that maintains a $(k, 2+\epsilon)$-bicriterion and uses $O(\epsilon^{-3} k \log n)$ space. Our algorithm works in three layers. The first layer 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 (in Sect. 4) maintains a prefix A that contributes at most an ϵ-small portion to OPT of the stream; this layer only requires the space needed to store the output of the first layer at two previous moments. The third layer (in Sect. 5) runs the facility location algorithm of [14], and at any moment only four instances of facility location are required to run in order to maintain the $1 - \frac{1}{n}$ probability guarantee.

Our techniques differ from previous facility-based algorithms in crucial ways. Like the previous works of [3,6], our algorithm operates in phases. However, the techniques used in these algorithms cause additional costs to be compounded during each phase. In contrast, our algorithm manages the stream so that we only incur cost during the two most recent phases. We avoid additional costs by maintaining a prefix A of the stream S such that $\mathrm{OPT}(A, k) \leq \epsilon \,\mathrm{OPT}(S)$ and such that we have an $O(1)$-approximate estimate of $\mathrm{OPT}(S)$ before processing the suffix $S \setminus A$.

Of course, it is impossible to have an $O(1)$-approximate estimate of $\mathrm{OPT}(S)$ before processing the suffix. However, Algorithm 1 allows to pretend that we have such an estimate. The fundamental idea is to always maintain the "next prefix" A'. If we ever detect that $\mathrm{OPT}(S)$ may have surpassed the upper bound of our estimate, then we replace the prefix A with A' and update the estimate accordingly.

Given an $O(1)$-approximate estimate of $\mathrm{OPT}(S)$, we can contruct a good approximation of $S \setminus A$ (see Sect. 3). Because $\mathrm{OPT}(A, k) \leq \epsilon \,\mathrm{OPT}(S)$, even a poor approximation on the prefix is sufficient. Combining both these pieces, we are able to maintain a low-constant solution over the stream.

3 Definitions

Our algorithm works for weighted sets of integral weight. For the bounds, let n be the total weight of the stream (the sum of the weights of each point in the stream). In fact, if n is not known in advance, a polynomial upper-bound will suffice. Note that n is assumed to be known in [3,6,15], so this does not add any additional restrictions. Let $(\mathcal{X}, d)$ be a metric space.

Definition 1 (Cost Function). *Given sets $A, C \subset \mathcal{X}$, the function $Cost(A, C)$ gives the cost of clustering A with center set C. Explicitly, $Cost(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c)$.*

Definition 2 (Optimum Cost). *The value* $\mathrm{OPT}(A, B, k)$ *is the lowest possible cost of clustering* A *with* k *centers from* B*. Explicitly,* $\mathrm{OPT}(A, B, k) = \min_{C \in B^k} \mathit{Cost}(A, C)$*. As shorthand,* $\mathrm{OPT}(A, k) = \mathrm{OPT}(A, \mathcal{X}, k)$ *where* $\mathcal{X}$ *is the entire metric space.*

Definition 3 (Connect Function). *Let* A, B *be multisets of equal weight.* $\mathit{Connect}(A, B)$ *is the minimum connection cost over all possible bijective maps* t *from* A *to* B*, where the connection cost of* t *is defined as* $\sum_{a \in A} d(a, t(a))$*.*

Definition 4 (Bicriterion). *An* (α, β)*-bicriterion approximation of the* k*-median clustering of* A *is a set* B *such that* $\mathrm{COST}(A, B) \leq \alpha\, \mathrm{OPT}(A, k)$ *and* $|B| \leq \beta k$*.*

We will make use of the following observation in Sect. 4.

Observation 1. *For any set* C *and equally weighted multisets* A *and* B*,* $\mathit{Cost}(A, C) \leq \mathit{Connect}(A, B) + \mathit{Cost}(B, C)$*.*

Proof. Let g be the optimal map from B to C. Let t be the optimal bijective map from A to B. Then 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 from A to C. The result followed by summing over all $a \in A$ and then noting that $d(a, h(a)) \leq d(a, g(t(a)))$.

The following observation is used in Sect. 6.

Observation 2. *If* $\mathit{Connect}(A_1, B_1) \leq v_1$ *and* $\mathit{Connect}(A_2, B_2) \leq v_2$*, then* $\mathit{Connect}(A_1 \cup A_2, B_1 \cup B_2) \leq v_1 + v_2$*.*

Proof. Let t_i be the optimal bijective map from A_i to B_i. Then consider $g(a) = t_i(a)$ if $a \in A_i$. Although 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.

4 Phase Manager

Over an insertion-only stream S, the algorithm of [3] maintains a multiset Q such that $\mathit{Connect}(S, Q) \leq \alpha\, \mathrm{OPT}(S, \mathcal{X}, k)$ from some constant α. We refer to this algorithm as **PLS** (which is the name of the algorithm in [6] that provides a similar guarantee). It constructs Q through a technique that connects points in S to other points in S and weights them accordingly. Therefore, we can make a simple modification to additionally maintain a value q such that $\mathit{Connect}(S, Q) \leq q$. For a section P of the stream, we denote the multiset Q by **PLS**(P) and we denote the value q by $q(P)$.

By running an offline γ-approximation for k-median on Q, we obtain a θ-approximation on the original stream where $\theta = 2\gamma(1 + \alpha)$. This is a standard result, and the reader is referred to [6] for details.

We denote the first N points of the stream by $[1, N]$. Our algorithm requires a monotonically increasing function $f([1, N])$ such that $\mathrm{OPT}([1, N], \mathcal{X}, k) \leq$

$f([1,N]) \leq \theta\ \mathrm{OPT}([1,N],\mathcal{X},k)$. We compute this function as follows. We define f' to be the sum of $q([1,N])$ and the cost of clustering **PLS**$([1,N])$ with its γ-approximation. By Observation 1, f' satisfies the desired inequalities, but f' may not be monotonically increasing because the γ-approximation may decrease at times. We define f recursively as $f([1,N]) = \max\{f'([1,N], f([1,N-1])\}$. Updating f requires $O(1)$ time and space because it is computed by taking the maximum of two already stored values. Now f is guaranteed to be monotonically increasing, and moreover is still satisfies the desired inequalities because $\mathrm{OPT}([1,N],\mathcal{X},k)$ is monotonically increasing.

Our algorithm relies on maintaining a partition of the stream into three segments. After processing S_N (the first N points from the stream), we write the elements of the filtration as A_N and B_N such that we have $\emptyset \subset A_N \subset B_N \subset S_N$. Here both A_N and B_N are prefixes of the stream, meaning that they are equal to $[1,m]$ for some $1 \leq m \leq N$. The following two loop invariants will be maintained, where $\beta = \alpha\theta/\epsilon$.

1. $f(A_N) \leq \beta^{-1} f(B_N)$
2. $f(B_N) > \beta^{-1} f(S_N)$

At the beginning of the stream, it will be necessary to establish the two loop invariants. We do this by letting B_m be the first $k+1$ distinct points and letting A_m be empty. Even if $k+1$ distinct points do not arrive until m is much greater than $k+1$, it is not difficult to see that this initialization procedure can be performed in $O(k \log n)$ memory.

Having established the loop invariants, Algorithm 1 maintains these invariants with a single instance of **PLS**. When a point arrives, it simply updates **PLS** and (if necessary) redefines the filtration to satisfy the invariants. Note that Algorithm 1 does not store any information besides the state of **PLS** for each element of the current filtration, resulting in memory requirement equal to that of **PLS**.

Algorithm 1. Update Process, upon arrival of point p_N

```
1: Update PLS with $p_N$ and compute $f(S_N)$
2: if $f(S_N) \geq \beta f(B_{N-1})$ then
3:     $A_N \leftarrow B_{N-1}$
4:     $B_N \leftarrow S_N$
5: else
6:     $A_N \leftarrow A_{N-1}$
7:     $B_N \leftarrow B_{N-1}$
```

Theorem 1. *Using $O(k \log n)$ memory, Algorithm 1 maintains a filtration $\emptyset \subset A_N \subset B_N \subset S_N$ such that $f(A_N) \leq \beta^{-1} f(B_N)$ and $f(B_N) > \beta^{-1} f(S_N)$.*

Proof. If the condition on Line 2 is not satisfied, then this implies that both invariants continue to hold. If the condition on Line 2 is satisfied, then the second invariant has been violated and must be re-established on Lines 3–4. We recursively assume that both invariants held for the filtration of S_{N-1}. The first invariant reads $f(A_N) \leq \beta^{-1} f(B_N)$ which is equivalent to $f(B_{N-1}) \leq \beta^{-1} f(S_{N-1})$; this is guaranteed to hold since the second invariant was violated. The second invariant reads $f(B_N) > \beta^{-1} f(S_N)$. Since on Line 4 we have $B_N = S_N$, this clearly holds for $\beta > 1$.

Algorithm 1 guarantees that when a phase change occurs, $\mathrm{OPT}(S, k)$ will remain within a constant multiplicative range before the next phase change. We now prove that this is the case.

Lemma 1. *Algorithm 1 guarantees that* $f(B_N)/\theta \leq \mathrm{OPT}(S_N, k) < \beta f(B_N)$.

Proof. The second inequality follows from the second loop invariant of Algorithm 1 and noting that $\mathrm{OPT}(S_N, k) \leq f(S_N)$. The first inequality follows from the approximation guarantee of f and monotonicity.

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. Other subroutines will observe (but not influence) Algorithm 1 and store two sets: $\mathbf{PLS}(A_N)$ and $\mathbf{PLS}(B_N)$.

5 Facility Manager

In this section, we present Algorithm 3 that will run in parallel with Algorithm 1. We will use a modified version of the online facility location algorithm of [14] as a subroutine. The main result of this section is Theorem 3 stating that Algorithm 3 maintains a weighted set Q_N such that $\mathit{Connect}(S_N \setminus A_N, Q_N) \leq (3 + \epsilon)\, \mathrm{OPT}(S_N, k)$ with high probability.

We recall the **OFL** Algorithm 2 with facility cost κ as used in [6]. We maintain a weighted set of facilities Φ, and denote $d(p, \Phi) = \min_{\phi \in \Phi} d(p, \phi)$, with $d(p, \emptyset) = \infty$ by convention. Upon receiving a point p, we open a weight $w(p)$ facility there with probability $w(p) d(p, \Phi)/\kappa$; otherwise we connect it to the nearest facility, incrementing that facilities weight by $w(p)$ and paying service cost $w(p) d(p, \Phi)$.

Algorithm 2. OFL(facility cost κ)

```
1: $ServiceCost \leftarrow 0$
2: $FacilityCount \leftarrow 0$
3: $\Phi \leftarrow \emptyset$
      **Update Process, upon receiving point $p_N$:**
4: **if** a probability $\min(1, w(p_N) d(p_N, \Phi)/\kappa)$ event occurs **then**
5:     Open a facility at $p_N$ with weight $w(p_N)$
6:     $FacilityCount \leftarrow FacilityCount + 1$
7: **else**
8:     Increment weight of a nearest facility to $p_N$ by $w(p_N)$
9:     $ServiceCost \leftarrow ServiceCost + w(p_N) d(p_N, \Phi)$
```

The following theorem follows from a tuning of parameters based on Theorem 3.1 of [3]. The original statement was for $\epsilon = 1$, so we include a sketch of how we modify their proof.

Theorem 2. *If* **OFL** *is run on a weighted set A of weight at most n using facility cost $\frac{L}{k(1+\log n)}$ where $L \leq \epsilon\,\mathrm{OPT}(A,k)$, then with probability at least $1 - \frac{1}{n}$ the service cost is at most $(2+7\epsilon)\,\mathrm{OPT}(A,k)$ and at most $7\epsilon^{-1}k(1+\log n)\frac{\mathrm{OPT}(A,k)}{L}$ facilities are opened.*

Proof (Proof Sketch). Consider an optimal center c that services the set $S \subset A$. Let $\varSigma$ be the total service cost of assigning S to c. For $j \geq 0$, define regions S_j 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 $\cup_{j>j'} S_j$ consists of at most a single point for $j' = \log_{1+\epsilon}(n/2) \leq 2\epsilon^{-1}\log n$ (whenever $\epsilon \leq 1/2$). As in the proof of [3], the service cost of all points after a facility is opened in a region is deterministically at most $(\frac{\epsilon}{1-\epsilon} + (1+\epsilon))\varSigma$. This follows by applying Markov's inequality to show the cost of connecting the nearest $\epsilon|S|$ points is at most $\frac{\epsilon}{1-\epsilon}\varSigma$.

As for before a facility opens, it is shown in [3] that the probability of having total service cost over x regions of at least y before a facility opens is at most $e^{x-y\frac{e-1}{e}}$. Here we now set $x = 2\epsilon^{-1}k(1+\log n)$ and $y = 2\frac{e}{e-1}\epsilon^{-1}k(1+\log n)$ to yield the result.

Algorithm 3 maintains a set of **OFL** instances, where n is the weight of the stream. After each phase change, it begins running $d+1$ instances of **OFL** with facility cost set to $\epsilon f(B_N)/\theta$. Run this instance until the end of the phase, and then increase the service cost and duplicate the instance $d+1$ times. At any moment, provide Q_N as the weighted set of facilities of the instance running in the bucket of the current phase with minimal service cost.

We will refer to an instance running in "bucket t". This is to avoid confusion because there will be instances running during phase t in buckets t and $t+1$. We discard buckets $t-1$ and earlier.

We now present the main theorem of this section.

Theorem 3. *With probability at least $1 - n^{-d}$, where d is a chosen parameter, Algorithm 3 maintains a weighted set Q_N such that Connect$(S_N \setminus A_N, Q_N) \leq (2+7\epsilon)(1+\epsilon)\,\mathrm{OPT}(S_N,k)$. The storage requirement is $O(d\epsilon^{-3}k\log n)$.*

Proof. The space bound is deterministic and follows easily from Algorithm 3. This is because Line 9 guarantees that we have at most $7\alpha\theta^2\epsilon^{-3}k(1+\log n)$ facilities per instance. We store at most $d+1$ instances in bucket $t+1$ and at most $d+1$ instances in bucket t, resulting in an overall storage of at most $14(d+1)\alpha\theta^2\epsilon^{-3}k(1+\log n)$ facilities.

We now prove that at least one instance in bucket t remained active throughout phase $t-1$ (by not opening too many facilities and thus terminating on line 9). Consider the instances in bucket t, which were first begun as a batch of $d+1$ instances at the beginning of phase $t-1$. Since Algorithm 1 shifts $A_N \leftarrow B_{N-1}$ at a phase change, these instances were started with facility cost

Algorithm 3. Update Process, upon receiving point p_N

```
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($\kappa$) in bucket $t+1$
7: else
8:     Update all running instances of OFL with point $p_N$
9:     Terminate instances with facility-count above $7\alpha\theta^2\epsilon^{-3}k(1+\log n)$
10: if bucket $t$ contains a running instance then
11:     $Q_N \leftarrow$ facilities of a bucket $t$ instance with minimal service cost
12: else
13:     $\Phi_2 \leftarrow$ facilities of a bucket $t+1$ instance with minimal service cost
14:     $Q_N \leftarrow \Phi_1 \cup \Phi_2$
```

$\epsilon f(A_N)/(\theta k(1+\log n))$ 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 we apply Theorem 2 to B'_N. By Theorem 1 we have $\mathrm{OPT}(B'_N, k)/(\epsilon f(A_N)/\theta) < \beta\theta\epsilon^{-1} = \alpha\theta^2\epsilon^{-2}$. With this bound, Theorem 2 guarantees with probability $1-n^{-d-1}$ that at least one of the $d+1$ instances will run on B'_N with at most $7\alpha\theta^2\epsilon^{-2}k(1+\log n)$ facilities. Since the number of facilities 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 $S_N \setminus B_N$ (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 $c = 3 + 7\epsilon$, let L_t (and similarly L_{t-1}) be $k(1+\log n)\kappa$ where κ is the facility cost used for instances in bucket t. We break into two cases and analyze each seperately. In the first case, suppose $\mathrm{OPT}(B'_N, k) \geq \epsilon\,\mathrm{OPT}(S_N, k)$. We repeat the previous analysis with Theorem 2 of running **OFL** on the segment $S_N \setminus A_N$ instead of $B'_N \setminus A_N$. Since $\mathrm{OPT}(S_N, k)/(\epsilon f(A_N)/\theta) \leq \epsilon^{-2}\,\mathrm{OPT}(B'_N, k)/(f(A_N)/\theta) < \alpha\theta^2\epsilon^{-3}$ and $L_t = \epsilon f(A_N)/\theta \leq \epsilon\,\mathrm{OPT}(A_N, k) < \epsilon\,\mathrm{OPT}(S_N, k)$, Theorem 2 gives a high-probability guarantee that an instance in bucket t has opened at most $7\alpha\theta^2\epsilon^{-3}k(1+\log n)$ facilities with service cost at most $c\,\mathrm{OPT}(S_N, k)$, and we are done. In the second case, suppose $\mathrm{OPT}(B'_N, k) < \epsilon\,\mathrm{OPT}(S_N, k)$. If there is an active instance in bucket t, then the connection cost is at most $c\,\mathrm{OPT}(S_N, k)$. If there are no active instances in bucket t, we return $\Phi_1 \cup \Phi_2$. We apply Theorem 2 to $B'_N \setminus A_N$ to show $\mathit{Connect}(B'_N \setminus A_N, \Phi_1) \leq c\,\mathrm{OPT}(B'_N, k)$. Line 3 implies $\mathit{Connect}(B'_N \setminus A_N, \Phi_1) = \mathit{Connect}(B_N \setminus A_N, \Phi_1)$, and therefore $\mathit{Connect}(B_N \setminus A_N, \Phi_1) < c\,\mathrm{OPT}(B'_N, k)$. $L_{t+1} = \epsilon f(B_N)/\theta \leq \epsilon\,\mathrm{OPT}(B_N, k) \leq \epsilon\,\mathrm{OPT}(S_N, k)$, and we apply the theorem again to $S_N \setminus B_N$ to show $\mathit{Connect}(S_N \setminus B_N, \Phi_2) \leq c\,\mathrm{OPT}(S_N, k)$. Then by Observation 2, we bound connection costs as $\mathit{Connect}(S_N \setminus A_N, \Phi_1 \cup \Phi_2) \leq c\,\mathrm{OPT}(B'_N, k) + c\,\mathrm{OPT}(S_N, k) < c(1+\epsilon)\,\mathrm{OPT}(S_N, k)$.

The above proof holds for a single phase. There is at least one point per phase, implying that there are at most n phases. Thus the $1 - n^{-d-1}$ probability guarantee for each phase becomes a $1 - n^{-d}$ probability guarantee over the stream.

From now on, we refer to the result of Theorem 3 as guaranteeing connection cost $(2+\epsilon)\,\mathrm{OPT}(S_N, k)$ instead of $(2+7\epsilon)(1+\epsilon)\,\mathrm{OPT}(S_N, k)$. This follows by selecting $\epsilon' = \epsilon/7$.

6 Combining Both Algorithms

Algorithm 1 provides a weighted set $\mathbf{PLS}(A_N)$ such that $\mathit{Connect}(A_N, \mathbf{PLS}(A_N)) \leq \alpha\,\mathrm{OPT}(A_N, k)$. Moreover, the algorithm guarantees that $f(A_N) \leq \beta^{-1} f(S_N)$, where $\beta = \alpha\theta/\epsilon$, and therefore $\mathrm{OPT}(A_N, k) \leq f(A_N) \leq \beta^{-1} f(S_N) \leq \beta^{-1}\theta\ \mathrm{OPT}(S_N, k) \leq \epsilon\alpha^{-1}\,\mathrm{OPT}(S_N, k)$. Together, this implies that $\mathit{Connect}(A_N, \mathbf{PLS}(A_N)) \leq \epsilon\,\mathrm{OPT}(S_N, k)$. Algorithm 3 provides us with a weighted set Q_N such that $\mathit{Connect}(S_N \setminus A_N, Q_N) \leq (2+\epsilon)\,\mathrm{OPT}(S_N, k)$. Define Σ_N as the union $\mathbf{PLS}(A_N) \cup Q_N$. This is the desired bicriterion since, by Observation 2, $\mathit{Connect}(S_N, \Sigma_N) \leq (2+2\epsilon)\,\mathrm{OPT}(S_N, k)$.

Our algorithm maintains Σ_N using $O(\epsilon^{-3} k \log n)$ space. Moreover, as shown above, Σ_N is a $(O(\epsilon^{-3} k \log n), 2+\epsilon)$-bicriterion for the stream S_N.

7 Appendix

7.1 Lower Bound

The following lower bound relies on the fact that the algorithm, if it uses sublinear space, must forget most of the input points. Missing a critical input point can prevent anything better than a 2-approximation existing among the points remaining in storage.

Theorem 4. *For the metric k-median problem, no* $\mathrm{polylog}(n)$*-space streaming algorithm can (with constant probability) maintain a (α, β)-bicriterion for $\beta < 2$.*

Proof. Consider a specific algorithm. Let $S(n)$ be the space-complexity of this algorithm, measured in the number of points able to be stored. Suppose that $S(n) \in o(n)$. Fix a value of n. Define $R(n) = \lceil \sqrt{nS(n)} \rceil$ and note that $R(n) \in o(n)$ We will construct an input for the 1-median case, and then show it can be modified for k-median. Let the input begin with $(p_1, \ldots, p_{R(n)})$ where $d(p_i, p_j) = 1 \forall i \neq j$ where $j \in \{1, \ldots, R(n)\}$. Thus the first $R(n)$ points are indistinguishable, so even for a non-deterministic algorithm there must exist a deterministic $c \in \{1, \ldots, R(n)\}$ such that after the algorithm passes the first $R(n)$ points, c is stored in memory with probability at most $S(n)/R(n) \leq \sqrt{S(n)/n} \in o(1)$. The entire input is then $(p_1, \ldots, p_{R(n)}, q_{R(n)+1}, \ldots, q_n)$ where $d(p_c, q_i) = 1 \forall i \in$

$\{R(n)+1,\dots,n\}$ and all other distances are given by the shortest path. Without p_c stored as a potential center, the next best clustering (using one of the first $R(n)$ points as the center) yields a cost of $R(n) - \alpha + 2(n - R(n))$ while the optimum (with p_c as the center) is $n-1$. Since α is a lower-bound on the storage requirement of the algorithm (it must at least store the bicriterion is provides), in the limit $n \to \infty$ this input has a cost-ratio approaching 2 with probability approaching 1. To extend to k-median, use the above input with size n/k and duplicate it k times (where each duplicate is at least distance 2 from any other). The value of c may be different for each of the k pieces, but there must exist a deterministic $(c_1,\dots,c_k)$ such that the above argument extends.

7.2 Computing the Constant of Previous Algorithms

In this section, we compute the constant of the approximation-algorithm in [6] which the authors leave unspecified. The lower-space algorithm of [3] has an even larger constant due to the high-probability guarantee on the facility location lemma of a $(3 + \frac{2e}{e-1})$-approximation instead of a 4-approximation.

In Sect. 2 of [6], the connection cost of the maintained PLS set is seen to be $\alpha = 4(1 + 4(\gamma + \beta))$. The constants γ and β are freely selected subject to the restraint that $\gamma+4(1+4(\gamma+\beta)) \le \gamma\beta$. By minimizing the function $\alpha(\gamma,\beta)$ subject to this constraint, we obtain a lower bound on the approximation-ratio of these algorithms. Since the function $\alpha(\gamma,\beta)$ has no critical points, its minimum must occur on the boundary of the constraint equation. Using Lagrange multipliers to minimize $\gamma + \beta$ subject to $\gamma + 4(1 + 4(\gamma + \beta)) = \gamma\beta$, we find $\gamma = 16 + \sqrt{276}$ and $\beta = \gamma + 1$, setting the final approximation ratio to over 1063.5.

References

1. Bādoiu, M., Har-Peled, S., Indyk, P.: Approximate clustering via core-sets. In: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, STOC 2002, pp. 250–257. ACM, New York (2002)
2. Bentley, J.L., Saxe, J.B.: Decomposable searching problems I. Static-to-dynamic transformation. J. Algorithms **1**(4), 301–358 (1980)
3. Braverman, V., Meyerson, A., Ostrovsky, R., Roytman, A., Shindler, M., Tagiku, B.: Streaming k-means on well-clusterable data. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, pp. 26–40. SIAM (2011)
4. Bury, M., Schwiegelshohn, C.: Random projections for k-means: maintaining coresets beyond merge & reduce. CoRR, abs/1504.01584 (2015)
5. Byrka, J., Pensyl, T., Rybicki, B., Srinivasan, A., Trinh, K.: 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 2015, pp. 737–756. SIAM (2015)
6. Charikar, M., O'Callaghan, L., Panigrahy, R.: Better streaming algorithms for clustering problems. In: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, STOC 2003, pp. 30–39. ACM, New York (2003)

7. Chen, K.: On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. SIAM J. Comput. **39**(3), 923–947 (2009)
8. Feldman, D., Langberg, M.: A unified framework for approximating and clustering data. In: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, STOC 2011, pp. 569–578. ACM, New York (2011)
9. Fichtenberger, H., Gillé, M., Schmidt, M., Schwiegelshohn, C., Sohler, C.: BICO: BIRCH meets coresets for k-means clustering. In: Bodlaender, H.L., Italiano, G.F. (eds.) ESA 2013. LNCS, vol. 8125, pp. 481–492. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40450-4_41
10. Guha, S.: Tight results for clustering and summarizing data streams. In: Proceedings of the 12th International Conference on Database Theory, ICDT 2009, pp. 268–275. ACM, New York (2009)
11. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: theory and practice. IEEE Trans. Knowl. Data Eng. **15**(3), 515–528 (2003)
12. Har-Peled, S., Kushal, A.: Smaller coresets for k-median and k-means clustering. Discrete Comput. Geom. **37**(1), 3–19 (2007)
13. Har-Peled, S., Mazumdar, S.: Coresets for k-means and k-median clustering and their applications. In: STOC 2004, pp. 291–300 (2004)
14. Meyerson, A.: Online facility location. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS 2001, p. 426. IEEE Computer Society, Washington, DC (2001)
15. Shindler, M., Wong, A., Meyerson, A.W.: Fast and accurate k-means for large datasets. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 24, pp. 2375–2383. Curran Associates Inc., Red Hook (2011)