Diameter and k-Center in Sliding Windows*

Vincent Cohen-Addad[1], Chris Schwiegelshohn[2], and Christian Sohler[2]

1 **Département d'informatique, École normale supérieure, Paris, France.**
vincent.cohen@ens.fr
2 **Efficient Algorithms and Complexity Theory, TU Dortmund, Germany.**
{firstname.lastname}@tu-dortmund.de

Abstract

In this paper we develop streaming algorithms for the diameter problem and the k-center clustering problem in the sliding window model. In this model we are interested in maintaining a solution for the N most recent points of the stream. In the diameter problem we would like to maintain two points whose distance approximates the diameter of the point set in the window. Our algorithm computes a $(3+\epsilon)$-approximation and uses $O(1/\epsilon \ln \alpha)$ memory cells, where α is the ratio of the largest and smallest distance and is assumed to be known in advance. We also prove that under reasonable assumptions obtaining a $(3-\epsilon)$-approximation requires $\Omega(N^{1/3})$ space.

For the k-center problem, where the goal is to find k centers that minimize the maximum distance of a point to its nearest center, we obtain a $(6+\epsilon)$-approximation using $O(k/\epsilon \ln \alpha)$ memory cells and a $(4+\epsilon)$-approximation for the special case $k=2$. We also prove that any algorithm for the 2-center problem that achieves an approximation ratio of less than 4 requires $\Omega(N^{1/3})$ space.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Streaming, k-Center, Diameter, Sliding Windows

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Analyzing big data sets from streams is a topic that has received considerable attention among theoretical and applied researchers. One of the most popular summarization and aggregation tasks studied in this context is to determine k clusters that represent key features of the input data with respect to certain properties.

In this paper we focus on variants of the k-center problem where we aim to find k points such that the maximum distance over all points to their closest center is minimized.

In the standard streaming setting, we constrain our algorithms to use as little space as possible while computing high-quality solutions. The complexity of clustering in general and k-center in particular is well understood for insertion-only streams where input points arrive one by one. The more general settings like dynamic streams and the sliding window model have also received some attention for other clustering objectives. Both models aim to incorporate dynamic behavior; in dynamic streams input points are removed via a dedicated delete operation and in the sliding window model older input expires as new elements arrive.

* Supported by Deutsche Forschungsgemeinschaft within the Collaborative Research Center SFB 876, project A2

Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper, we consider a fixed window size consisting of N points, which is the most widely studied variant of this model in theoretical computer science, but our algorithms also work in the case that the maximum number of points within the window is a function of time.

Our Contribution

The metric diameter problem is to find two points of maximum distance among a set of points lying in some metric space. For this problem, we give a $(3+\varepsilon)$-approximation algorithm in the sliding window model that stores $O(\frac{1}{\varepsilon}\log\alpha)$ points, where $\alpha = \frac{\max_{p,q}\operatorname{dist}(p,q)}{\min_{p,q}\operatorname{dist}(p,q)}$ is the ratio of largest and smallest possible distance between the points. This is a substantial improvement over the best and to our knowledge only sliding window algorithm for diameter in general metric spaces by Chan and Sadjad [8] which computes a $(2^{m+2}-2+\varepsilon)$ approximation with $O(N^{1/(m+1)}\log\alpha)$ points for any $m > 0$.

Under reasonable assumptions (which are common for our model of computation), we obtain a lower bound of $\Omega(\sqrt[3]{N})$ for any algorithm achieving a $3-\varepsilon$ approximation to the diameter problem for any $\varepsilon > 0$.

To our knowledge there exists no previous work on k-center in sliding windows. For 2 centers our diameter algorithm yields a $(4+\varepsilon)$-approximate clustering. Under the aforementioned assumptions, we are also able to obtain a matching lower bound. For arbitrary values of k, we are able to obtain a $(6+\varepsilon)$-approximate algorithm using $O(k/\varepsilon\log\alpha)$ points in metric spaces.

Techniques

The popular histogram approaches introduced by Datar et al. [12] and Braverman and Ostrovsky [6] do not seem to be applicable to max-norm objectives such as diameter and k-center. Our algorithm for the diameter (see Section 3) aims to find for each estimate of the value γ of the diameter two certificate points with distance greater than γ, while maintaining the two most recent points close to the two points forming the certificate. With every additional input point, we check whether we are able to update the certificate to a more recent timestamp.

For our lower bounds, we utilized the fact that any algorithm working in the metric distance model is restricted to storing only input points. For deterministic algorithms, we are then able to insert an appropriately hard instance based on the points forgotten by the input. For randomized algorithms, we add additional points in which we hide a hard, randomly chosen instance for deterministic algorithms. A more in-depth description of our approach as well as a discussion on the generality of our results can be found in Section 5.

The analysis of the 2-center algorithm is similar to that of the popular 2-approximation by Gonzales [14] and Hochbaum and Shmoys [16]. Since it does not yield an approximation of the optimum value, this technique seems difficult to generalize for larger values of k.

Related Work

Diameter

Feigenbaum et al. [13] were the first to consider the diameter in the sliding window model. For d dimensions in Euclidean space, their algorithm uses $O((\frac{1}{\varepsilon})^{(d+1)/2}\log^3 N(\log\alpha+\log\log N+\frac{1}{\varepsilon}))$ bits of space. They also give a lower bound of $\Omega(\frac{1}{\varepsilon}\log N\log\alpha)$ for a $(1+\varepsilon)$ approximation factor in one dimension and, implicitly, a $\Omega(\log\alpha)$ space bound for any multiplicative

approximation factor. This lower bound was later matched by Chan and Sadjad [8], who also gave an improved space bound of $O(\left(\frac{1}{\varepsilon}\right)^{(d+1)/2} \log \frac{\alpha}{\varepsilon})$ points for higher dimensions. For more general metric spaces, they obtain a $(2^{m+2} - 2 + \varepsilon)$ approximation with $O(N^{1/(m+1)})$ points.

In the metric distance oracle model (formally defined in Section 2) there exists a folklore 2 approximation that maintains the first point p and the point with maximum distance from p. Guha [15] showed this algorithm to be essentially optimal, as no algorithm storing less than $\Omega(n)$ points can achieve a ratio better than $2 - \varepsilon$ for any $\varepsilon > 0$. For Euclidean spaces, the best streaming algorithm with a polynomial dependency on d is due to Agarwal and Sharathkumar [2] with an almost tight approximation ratio of $\sqrt{2} + \varepsilon$ in $O(d\varepsilon^{-3} \log(1/\varepsilon))$ space. Agarwal et al. [1] proposed a $(1+\varepsilon)$-approximation using $O(\varepsilon^{-(d-1)/2})$ points. Similar space bounds seem likely for dynamic streams although none have been published to our knowledge. For large d, Indyk [17] gave a sketching scheme with approximation factor $c > \sqrt{2}$ and space $O(dn^{1/(c^2-1)} \log n)$.

k-Center

In one of the earliest works on clustering in streams, Charikar et al. [9] gave a number of incremental clustering algorithms for metric k-center, among other results. While storing no more than $k + 1$ points at any given time, they were able to derive a deterministic 8 approximation and a randomized $2e \approx 5.437$ approximation. They also show that no incremental algorithm can be better than 3. McCutchen and Khuller [19] and Guha [15] independently derived a $(2 + \varepsilon)$-approximate algorithm using $O(k/\varepsilon \log 1/\varepsilon)$ space, with Guha giving an almost tight lower bound of $\Omega(n)$ space for any algorithm achieving a better approximation ratio than 2. In their paper, McCutchen and Khuller [19] also studied the problem with z outliers, giving a $(4 + \varepsilon)$ approximate algorithm that stores $O(\varepsilon^{-1}kz)$ points, see also Charikar et al. [10] for an earlier treatment of the problem. Further improvements are possible in Euclidean spaces. Zarrabi-Zadeh showed how to maintain ε-coresets in streams using $O(k\varepsilon^{-d})$ points for k-center [21]. For small values of k, Kim and Ahn [18] were able to break the 2 barrier without having an exponential dependency on d, giving a $1.8 + \varepsilon$ approximation while storing $O(2^k(k+3)!\varepsilon^{-1})$ points. The special case of $k = 1$ for Euclidean distances also known as the minimum enclosing ball problem is one of the most extensively studied topics in streaming literature. We only review the best known bounds. Zarrabi-Zadeh gave an insertion-only algorithm storing $O(\varepsilon^{-(d-1)/2} \log 1/\varepsilon)$ points [22]. Agarwal and Sharathkumar [2] showed that no algorithm with polynomial dependency on d can achieve a better approximation ratio than $(1 + \sqrt{2})/2 \approx 1.207$ and provided an algorithm which after a re-analysis by Chan and Pathak [7] is now known to give a 1.22 approximation with space roughly $O(d)$.

We are not aware of any work on k-center in sliding windows, though Babcock et al. [3] and more recently Braverman et al. [4, 5] gave a $O(1)$ approximation for metric and a $(1+\varepsilon)$ approximation for Euclidean k-median and k-means problems. The related problem of cut sparsification has also received some attention, see Crouch et al. [11].

The structure of the paper is as follows. Section 2 introduces the model and the definitions. Section 3 is dedicated to the description and analysis of our algorithm for the diameter problem. The analysis and description of our algorithm for the k-center problem is in Section 4. Finally, Section 5 contains the lower bounds for both the diameter and k-center problems.

2 Preliminaries

Let (A, dist) be a metric space where A is a set of points and $\text{dist} : A \times A \mapsto R_+$ is a distance function. A stream is a (potentially infinite) sequence of points from the metric space A (note that a point can be appear multiple times in the stream). The sliding window of size N contains the most recent N elements of the stream.

We introduce the *Time To Live* value of a point p: Upon insertion $TTL(p)$ is set to the window size N and with each subsequently inserted point it is decremented. We say that p *expires* if $TTL(p) = 0$. We extend the common use of TTL to negative numbers to indicate the number of points submitted after expiration, i. e., $TTL(p) = -10$ means that 10 points were submitted after the expiration of p. We define the aspect ratio $\alpha = \frac{\max_{p,q \in A} \text{dist}(p,q)}{\min_{p,q \in A} \text{dist}(p,q)}$. To query the distance between two points p and q, we invoke a distance oracle $\text{dist}(p, q)$. We assume that the oracle can accessed only for those points we currently keep in memory and that the oracle itself requires no additional space.

Algorithm 1 Sliding Window Algorithm for $(\gamma, 3 \cdot \gamma)$-gap Diameter

```
1:  c_old, q, r ← first point of the stream;
2:  c_new ← null;
3:  for all element p of the stream do
4:      if certificate point c_old expires then
5:          if (c_new ≠ null ∧ c_old = q) then
6:              c_old ← r; c_new ← null;
7:          if (c_new ≠ null ∧ c_old ≠ q) then
8:              c_old ← q; c_new ← null;
9:          if c_new = null then
10:             c_old ← r;
11:     INSERT(p);
12:     r ← p;
13: procedure INSERT(p)
14:     if c_new = null then
15:         if dist(p, r) > γ then
16:             c_old, q ← r; c_new ← p;
17:         else if dist(p, c_old) > γ then
18:             q ← r; c_new ← p;
19:     else
20:         if dist(p, r) > γ then
21:             c_old, q ← r; c_new ← p;
22:         else if dist(p, c_new) > γ then
23:             c_old ← c_new; q ← r; c_new ← p;
24:         else if dist(p, q) > γ then
25:             if c_old ≠ q then
26:                 c_old ← q; q ← r; c_new ← p;
```

3 The Metric Diameter Problem

For a given estimate γ of the diameter, our algorithm for the metric diameter problem either produces two witness points at distance greater than γ or a point c that has a certain degree of centrality among the points in the current window. More formally, all points of the window inserted up to the insertion time of c will be proven to have distance at most 2γ from one another and points inserted after c will have distance at most γ from c. Thus, the diameter is at most 3γ.

Specifically, Algorithm 1 aims at maintaining a certificate for the diameter consisting of two points c_{old} and c_{new} such that $\text{dist}(c_{old}, c_{new}) > \gamma$ and $TTL(c_{old}) < TTL(c_{new})$. In addition, we also store the point q submitted immediately prior to c_{new} and the most recent point r. When a new point arrives, we test whether, based on the points we currently keep in memory, we can produce two points each with a larger TTL than $TTL(c_{old})$ with distance more than γ. If we find such a pair, we update the points accordingly, if not we update r

and possibly q.

The algorithm has two different states depending on whether it found a pair of points of distance more than γ or not. The first state is indicated by $c_{new} = \textbf{null}$ and corresponds to the case that no such pair of points has been found. In this case, the algorithm maintains the following invariant, which certifies that the diameter of the points in the sliding window is at most $3 \cdot \gamma$.

We first observe that c_{old} is always inside the sliding window.

▶ **Invariant 1.** If $c_{new} = \textbf{null}$, the following statements hold:

a) For any points a, b with $0 \leq TTL(a), TTL(b) \leq TTL(c_{old})$, we have $\text{dist}(a, b) \leq 2 \cdot \gamma$.
b) For any point a with $TTL(a) > TTL(c_{old})$, we have $\text{dist}(a, c_{old}) \leq \gamma$.

The second state corresponds to the case that we discover two points c_{old} and c_{new} with distance more than γ and is indicated by $c_{new} \neq \textbf{null}$. Besides the obvious invariant that $TTL(c_{new}) > TTL(c_{old})$, we also have to maintain the following technical invariants that are required for a new assignment of c_{old} when it expires from the window.

▶ **Invariant 2.** If $c_{new} \neq \textbf{null}$ then the following statements hold:

a) $\text{dist}(c_{old}, c_{new}) > \gamma$.
b) For any point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, we have $\text{dist}(a, c_{old}) \leq \gamma$.
c) For any point a with $TTL(c_{new}) < TTL(a)$, we have $\text{dist}(a, c_{new}) \leq \gamma$.
d) If $c_{old} \neq q$ then for any point a with $TTL(q) < TTL(a)$, we have $\text{dist}(a, q) \leq \gamma$.

We observe that all the invariants hold initially, i.e. before line 3 of the algorithm is executed the first time. We also observe that Invariants 2a)-d) only apply to points that appear after c_{old}. It suffices to focus on these points because we only change c_{old} to points that arrive later and so we maintain our certificate at least until the time when c_{old} expires (and so all earlier points are gone).

▶ **Lemma 1.** *If $c_{new} = \textbf{null}$ and Invariant 1 is satisfied before* Insert *then one of the following statements holds:*

1) If $c_{new} = \textbf{null}$ after line 12 then Invariant 1 is satisfied.
2) If $c_{new} \neq \textbf{null}$ after line 12 then Invariant 2 is satisfied.

Proof. We never execute lines 19-26 of Insert. If $\text{dist}(p, r) > \gamma$ then $c_{new} \neq \textbf{null}$ and Invariant 2.a) holds due to line 16, Invariant 2.b) holds due to the fact that there exists no point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, Invariant 2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2.d) holds due to $c_{old} = q$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{old}) > \gamma$ then $c_{new} \neq \textbf{null}$, and Invariant 2.a) holds due to line 18, Invariant 2.b) holds due to Invariant 1.b), Invariant 2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2.d) holds due to $r = q$ (before line 12) and line 15. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{old}) \leq \gamma$ then Invariant 1 continues to be satisfied for all points with TTL smaller than p and for the point p due to line 17. ◀

▶ **Lemma 2.** *If $c_{new} \neq \textbf{null}$ and Invariant 2 is satisfied before* Insert*, then Invariant 2 is satisfied after line 12.*

Proof. If $\text{dist}(p, r) > \gamma$ then Invariant 2.a) holds due to line 21, Invariant 2.b) holds due to the fact that there exists no point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, Invariant 2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2.d) holds due to $c_{old} = q$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{new}) > \gamma$ then Invariant 2.a) holds due to line 23, Invariant 2.b) and 2.d) hold due to Invariant 2.c) before Insert, and Invariant 2.c)

holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$. If $\text{dist}(p,r) \leq \gamma$ and $\text{dist}(p,c_{new}) \leq \gamma$, and $\text{dist}(p,q) > \gamma$ and $q \neq c_{old}$ then Invariant 2.a) holds due to line 26, Invariant 2.b) holds due to Invariant 2.d) before INSERT, Invariant 2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2.d) holds due to $q = r$ and line 20. If $\text{dist}(p,r) \leq \gamma$ and $\text{dist}(p,c_{new}) \leq \gamma$, and $\text{dist}(p,q) > \gamma$ and $q = c_{old}$ or $\text{dist}(p,q) \leq \gamma$, the Invariants 2.a) - d) hold for all points except for the newest, for which the invariants hold due to lines 20, 22, 24 and 25. ◀

▶ **Lemma 3.** *If $c_{new} =$ **null** and Invariant 1 is satisfied before the line 4, then it is satisfied before INSERT (line 11).*

Proof. If c_{old} does not expire, then the claim obviously holds. Otherwise we execute line 10. Let c'_{old} be the expired point. Then we have for any two points a, b with $TTL(c'_{old}) < TTL(a), TTL(b) \leq TTL(r)$ $\text{dist}(a,c'_{old}), \text{dist}(b,c'_{old}) \leq \gamma$ due to Invariant 1.b) before line 4 and hence $\text{dist}(a,b) \leq 2\gamma$. Invariant 1.b) follows from $c_{old} = r$. ◀

▶ **Lemma 4.** *If $c_{new} \neq$ **null** and Invariant 2 is satisfied before the line 4, then one of the following statements holds before INSERT (line 11):*

*1) $c_{new} =$ **null** and Invariant 1 is satisfied.*
*2) $c_{new} \neq$ **null** and Invariant 2 is satisfied.*

Proof. If c_{old} does not expire the second claim immediately holds. Otherwise, we denote by c'_{old} the expired point. If $c'_{old} = q$, then we execute line 6. We set $c_{old} = r$, naturally satisfying Invariant 1.b). Further, at this time we have $TTL(c_{new}) = 1$, and for any two points a, b with $TTL(c_{new}) \leq TTL(a), TTL(b) \leq TTL(r)$ we have $\text{dist}(a,c_{new}), \text{dist}(b,c_{new}) < \gamma$ due to Invariant 2.c) before line 4 and hence $\text{dist}(a,b) \leq 2\gamma$, satisfying Invariant 1.a).

If $c'_{old} \neq q$, then we execute line 8. We set $c_{old} = q$. For any two points a, b with $TTL(c'_{old}) < TTL(a), TTL(b) < TTL(c_{new})$ $\text{dist}(a,c'_{old}), \text{dist}(b,c'_{old}) \leq \gamma$ due to Invariant 2.b) before line 4 and hence $\text{dist}(a,b) \leq 2\gamma$, satisfying Invariant 1.a). For all points a with $TTL(a) > q$, Invariant 1.b) after line 12 follows from Invariant 2.d) before line 4. ◀

The proof of the theorem is a direct consequence of the invariants but included for completeness.

▶ **Theorem 5.** *Given a set of points A with aspect ratio α and a window of size N, there exists an algorithm computing a $3(1+\epsilon)$-approximate solution for the metric diameter problem storing at most $8/\epsilon \cdot \ln \alpha$ points. The update time per point is $O(\varepsilon^{-1} \log \alpha)$.*

Proof. For any given estimate γ, we either have two points at distance at least γ, or Invariant 1 holds. In the latter case, we can bound the maximum diameter of two points p and q via the following case analysis.
$\boldsymbol{TTL(p), TTL(q) \leq TTL(c_{old})}$**:** Then $\text{dist}(p,q) \leq 2\gamma$.
$\boldsymbol{TTL(p) \leq TTL(c_{old}) < TTL(q)}$**:** Then $\text{dist}(p,q) \leq \text{dist}(p,c_{old}) + \text{dist}(c_{old},q) \leq 3\gamma$.
$\boldsymbol{TTL(c_{old}) < TTL(p), TTL(q)}$**:** Then $\text{dist}(p,q) \leq \text{dist}(p,c_{old}) + \text{dist}(c_{old},q) \leq 2\gamma$.
Now define an exponential sequence to the base of $(1+\varepsilon)$, such that any value between $\min \text{dist}(p,q)$ and $\max \text{dist}(p,q)$ is $(1+\varepsilon)$ approximated. For each power of $(1+\varepsilon)$, we run Algorithm 1. Let γ be the largest value for which one of the instances of Algorithm 1 returns two points. The next larger estimate $\gamma \cdot (1+\varepsilon)$ guarantees us no diameter of size $3(1+\varepsilon)\gamma$, proving an approximation guarantee of at most $\frac{3(1+\varepsilon)\gamma}{\gamma} = 3(1+\varepsilon)$ The memory usage of the algorithm consists of 4 points per instance of Algorithm 1 and $\log_{1+\varepsilon} \alpha = \frac{\ln \alpha}{\ln(1+\varepsilon)} \leq \frac{2}{\varepsilon} \ln \alpha$ instances. ◀

▶ Remark. To adapt this algorithm for windows where the maximum number of points are time dependent (e. g., the diameter of all points seen in the last hour) rather than the last N points, we can simply decouple the insertion procedure from the deletion routine. Whenever a point we currently keep in memory expires, we execute lines (4-10) and whenever a new point arrives, we call the INSERT procedure and line 12. Neither the invariants nor the proofs are affected in any way by this change.

4 The k-Center Problem

A 4-Approximation for Metric 2 Center

We run Algorithm 1 and show that, in the case of $k = 2$, it outputs a solution of cost at most 4 times the optimal solution. More precisely, let γ be the smallest estimate such that Algorithm 1 produces one point c with $\text{dist}(q, c) \leq 2\gamma$ for any point q in the current window. Further let a and b be the two points at distance greater than $\frac{\gamma}{1+\varepsilon}$ outputted by Algorithm 1 for the next smaller estimate. W.l.o.g let $\text{dist}(a, c) \geq \text{dist}(b, c)$. Then $\{a, c\}$ form a 4 approximation.

▶ **Theorem 6.** *Given a set of points A with aspect ratio α and a window of size S, there exists an algorithm computing a $4(1+\epsilon)$-approximate solution for the 2-center problem storing at most $8/\epsilon \cdot \ln \alpha$ points. The update time per point is $O(\varepsilon^{-1} \log \alpha)$.*

Proof. For c, the conditions of Invariant 1 apply, i.e. for any point p in our current window, we have $\text{dist}(p, c) \leq 2\gamma$. We now distinguish between two cases.

OPT $\geq \frac{\gamma}{2(1+\varepsilon)}$: We have $\text{dist}(p, \{a, c\}) \leq \text{dist}(p, c) \leq 2\gamma \leq 4 \cdot (1 + \varepsilon) \cdot \text{OPT}$.

OPT $< \frac{\gamma}{2(1+\varepsilon)}$: We first observe that a and b each fall into distinct clusters as their pairwise minimum distance is at least $\frac{\gamma}{1+\varepsilon}$. If a and c lie in distinct clusters, we have a 2-approximate solution, so we assume this not be the case. Then $\text{dist}(a, c) \leq 2 \cdot \text{OPT}$ and by construction, $\text{dist}(a, c) \geq \text{dist}(b, c)$. Then for any point p in the same cluster as b we have $\text{dist}(p, b) \leq 2 \cdot \text{OPT}$ and hence $\text{dist}(p, c) \leq \text{dist}(p, b) + \text{dist}(b, c) \leq 2 \cdot \text{OPT} + 2 \cdot \text{OPT} = 4 \cdot \text{OPT}$.

The proof of the space bound is analogous to that of Theorem 5. ◀

6-Approximation for Metric k-Center

A high level description of our algorithm is as follows, see also Algorithm 2 for pseudocode. We maintain a set A of at most $k+1$ *attraction points*. For each attraction point a, we maintain the newest point $R(a)$ within radius 2γ as a *representative*, i.e. $R(a) = \underset{p:\ \text{dist}(p,a)\leq 2\gamma}{\text{argmax}} TTL(R(a))$. When an attraction point expires, the representative point remains in memory. Call the set of representative points whose attraction points expired, the *orphaned representatives* O, and the set of representative points whose attraction points are still in the current window *active representatives* R. A new point p may become an attraction point if its distance is greater than 2γ to any point in A upon insertion. If the cardinality of A is greater than k, we retain the newest $k+1$ attraction points of A and all points with a greater TTL than the minimum TTL of A.

When asked to provide a clustering, we iterate through all estimates and either provide a counter example, or find a clustering which is then guaranteed to be a $6(1+\varepsilon)$-approximation. Our set of centers C first consists of an arbitrarily chosen point $p \in A \cup R \cup O$. Thereafter we greedily add any point point $q \in A \cup R \cup O$ with distance $\text{dist}(q, C) > 2\gamma$. If upon

termination $|C| > k$, we have a certificate for OPT $> \gamma$ and move to the next higher estimate. The smallest estimate with $|C| \leq k$ is then guaranteed to be a 6 approximation.

We start by giving the space bound.

Algorithm 2 Sliding Window Algorithm for $(\gamma, 6 \cdot \gamma)$-gap k-Center

```
1:  A, R, O ← ∅;
2:  for all element p of the stream do
3:      if q ∈ O expires then
4:          O ← O \ {q};
5:      if a ∈ A expires then
6:          DeleteAttraction(a);
7:      Insert(p);
8:  procedure DeleteAttraction(a)
9:      O ← O ∪ {R(a)};
10:     R ← R \ {R(a)};
11:     A ← A \ {a};
12: procedure Insert(p)
13:     D ← {a ∈ A | dist(p, a) ≤ 2 · γ};
14:     if D = ∅ then
15:         A ← A ∪ {p}
16:         R(p) ← p
17:         R ← R ∪ {R(p)}
18:         if |A| > k + 1 then
19:             a_old ← argmin_{a∈A} TTL(a);
20:             DeleteAttraction(a_old);
21:         if |A| > k then
22:             t ← min_{a∈A} TTL(a);
23:             for all q ∈ O do
24:                 if TTL(q) < t then
25:                     O ← O \ {q};
26:     else
27:         for all a ∈ D do
28:             Exchange R(a) with p in R;
```

▶ **Lemma 7.** *At any given time, the number of points kept in memory is bounded by at most* $3(k+1)$.

Proof. We number all attraction points we keep in memory via the sequence in which they arrived, i.e. a_1 is the first attraction point, a_2 the second, etc. Call this sequence S. Note that in this sequence a_1 also expires before a_2.

At any given time, we maintain at most $k+1$ attraction points A and $k+1$ active representative points R due to lines 18-20 and the subroutine DeleteAttraction (lines 8-11). What remains to be shown is that the number of orphaned representative points O also never exceeds $k+1$.

First, we show that $TTL(a_{i+k+1}) > TTL(R(a_i)) \geq TTL(a_i)$. We distinguish between two cases. If a_i expires, then a_{i+k+1} gets inserted after a_i exits the window, hence $TTL(a_{i+k+1}) > N+1+TTL(a_i)$ and $TTL(R(a_i))+N \leq TTL(a_i)$. Otherwise, a_i gets deleted via lines 18-20 in the exact same time step in which a_{i+k+1} got inserted, in which case the claim also holds.

Now consider any point of time and let j be the maximum index of any attraction point in S that has expired. By the above reasoning, any representative spawned by $a_{j-(k+1)}$ is no longer in memory, and the space bounds holds. ◀

▶ **Lemma 8.** *Let P be a set of points in a given window, $\gamma > 0$ an estimate of the clustering cost, $A \cup R \cup O$ the set of points we currently keep in memory with $|A| \leq k$. Then* $\max_{q \in P} \text{dist}(p, R \cup O) \leq 4\gamma$.

Proof. We note that for any attraction point a, the representative $R(a)$ has maximum TTL among all points with distance at most 2γ. When a point p arrives, it has distance at most 2γ to some attraction point (which may be identical to p if we create a new one). Hence, if $R(a)$ is still in memory, the claim holds for p.

We now argue that by executing lines 18-25, all points p with $\operatorname{dist}(p, R \cup O) > 4\gamma$ have $TTL(p) < \min_{a \in A} TTL(a)$. If $TTL(p) > \min_{a \in A} TTL(a)$, then there exists an attraction point a' such that $\operatorname{dist}(p, a') \leq 2\gamma$. Then we have $TTL(R(a')) \geq TTL(p) > \min_{a \in A} TTL(a)$ and $\operatorname{dist}(p, R(a')) \leq 4\gamma$. Due to lines 24-25, $R(a')$ is never deleted until it expires. ◀

Combining these lemmas and using arguments analogous to those of the proof of Theorem 5, we have:

▶ **Theorem 9.** *Given a set of points P with aspect ratio α and a window size N, there exists an algorithm computing a $6(1+\epsilon)$-approximate solution for the metric k-center problem storing $6(k+1)\ln(\alpha)/\epsilon)$ points. The update time per point is $O(k^2\varepsilon^{-1}\log\alpha)$.*

Proof. Again define an exponential sequence to the base $(1+\varepsilon)$ and run Algorithm 2 in parallel for all powers of $(1+\varepsilon)$ as objective value estimates. The space bound then follows from Lemma 7.

For each estimate γ, we greedily compute a clustering of $A \cup R \cup O$ where the pairwise distance between centers is greater than 2γ. Now consider the smallest estimate γ' for which the greedy clustering requires at most k centers C.

We have $\max_{p \in A \cup R \cup O} \operatorname{dist}(p, C) \leq 2\gamma'$. We further have for any point q in the current window $\max_{q \in P} \operatorname{dist}(q, C) \leq \max_{q \in P} \operatorname{dist}(q, R \cup O) + \max_{p \in A \cup R \cup O} \operatorname{dist}(p, C) \leq 4\gamma' + 2\gamma' \leq 6\gamma'$ due to Lemma 8. Since we have $\mathrm{OPT} > \frac{\gamma'}{1+\varepsilon}$, C is a $6(1+\varepsilon)$ approximation. ◀

5 Lower Bounds

Our lower bounds for the studied problems hold for the metric oracle distance model. Whenever we wish to know the distance between two points p, q, we have to store the points in their entirety in order to invoke the oracle. The fundamental assumption used in the proofs of this section is that the algorithm cannot create new points, unlike, for instance, in Euclidean spaces, where we can store projections, means and similar points. In particular, this implies that once a point is discarded by the algorithm, it cannot be recalled by any means at a later date. Without any assumptions as to how the points are encoded, we measure the space complexity of an algorithm via the number of stored points, rather than the number of bits. We do not consider the space required to store the distance oracle, the TTL of each point or any other information we might wish to store. A similar reasoning can be also found in the paper by Guha [15], where the author was able to derive a lower bound of $\Omega(k^2)$ points for any deterministic single-pass streaming algorithm approximating the cost of the optimal k-center clustering up to a factor $2 + 1/k$.

We first describe an adversarial input sequence for deterministic algorithms for the diameter problem and give a proof for randomized algorithms. With some modification, these ideas can be extended to the k-center problem, as well. We aim for a lower bound of $\Omega(\sqrt{N})$ points for deterministic algorithms. We divide the input into $\sqrt{N}$ buckets containing $\sqrt{N}$ points each. Unless the distances between two points are further specified, we will set these distances to 1. Denote the ith bucket by B_i and the jth point of bucket B_i by

$p_{i,j}$ with $i, j \in \{0, \ldots, \sqrt{N} - 1\}$. The points appear bucket by bucket, that is, $p_{i,j}$ is the $i \cdot \sqrt{N} + j + 1$th input point.

We assume that an algorithm always stores less than $\sqrt{N}$ points. Therefore, the algorithm must discard at least one point of bucket B_i before reading the first point of bucket B_{i+1}. Let f_i be such a discarded point in B_i. To any point from some bucket B_j, $j > i$, we then set the distance to f_i to be 2. By the same reason, there is at least one bucket without any stored points when the $N+1$st input point is read. Let B_t be this bucket. We now introduce the $N+1$st input point p that satisfies the distances $\operatorname{dist}(p, p_{i,j}) = 1$ if $i > t$, $\operatorname{dist}(p, p_{i,j}) = 3$ and $\operatorname{dist}(p, p_{i,j}) = 2$ otherwise.

We proceed to insert copies of p until all points in buckets B_i with $i < t$ are expired. Therefore, there is no pair of points in memory with distance larger than 1. The algorithm can only output two points at distance 1 whereas the true diameter is 3.

▶ **Theorem 10.** *For windows of size N, any deterministic sliding window algorithm outputting a solution of cost greater than $\frac{1}{3}OPT$ for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt{N})$ points.*

Recall that the algorithm by Chan and Sadjad [8] achieves a $2^{m+2} - 2 + \varepsilon$ approximation using $O(N^{1/(m+1)} \log \alpha)$ points, and it also falls under the same computational restrictions for the algorithms of this lower bound. Therefore, this lower bound cannot be strengthened by much, as their algorithm achieves a better approximation than 3 using roughly $N^{0.76}$ points by setting $m < \log 5/4$.

To utilize this instance for randomized algorithms, we require two modifications. First, we add additional points per bucket and uniformly choose f_i such that a randomized algorithm has little chance of retaining the correct point per bucket. Second, we use p (and its copies) to uniformly select a bucket B_t which the algorithm will have discarded with good probability.

▶ **Theorem 11.** *For windows of size N, any randomized sliding window algorithm outputting a solution of cost greater than $\frac{1}{3}OPT$ with probability greater than $\frac{1}{2}$ for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt[3]{N})$ points.*

Proof. For ease of exposition, we will use a window of size $\Theta(N)$. The theorem then follows by rescaling N. We use $4N^{1/3}$ buckets consisting of $32N^{2/3}$ points each. In the following, any distance that is not further specified is set to be 1.

We iteratively replace one randomly chosen point from bucket B_i with f_i where f_i has distance 2 to any point from bucket B_j with $j > i$. At the end of the stream, we insert a point p with the following distances. First, choose a random bucket B_t and set $\operatorname{dist}(p, f_t) = 3$ and $\operatorname{dist}(p, q) = 2$, where $q \in B_t \setminus \{f_t\}$. Any point inserted after bucket B_t has distance 1 to p and any point inserted before B_t has distance 2. We then repeatedly add copies of p at total of N times.

To show that the distances still satisfy the triangle inequality, we first observe that only $\operatorname{dist}(p, f_t)$ is neither 1 or 2 and thus requires special consideration. Here, we have $3 = \operatorname{dist}(p, f_t) \leq \operatorname{dist}(p, q) + \operatorname{dist}(q, f_t) = 1 + 2$ for $q \in B_i$ with $i > t$, and $3 = \operatorname{dist}(p, f_t) \leq \operatorname{dist}(p, q) + \operatorname{dist}(q, f_t) \leq 2 + 1$ for $q \in B_i$ with $i \leq t$.

At any given time, the algorithm has to output a pair of points whose distance is within a factor 3 of the diameter of the current window. Observe that if the algorithm did not store any of the replaced points $\{f_0, \ldots, f_{4N^{1/3}-1}\}$ and not any point of bucket B_t then the algorithm is not able to produce two points at distance greater than 1. Hence, by Yao's minimax principle, it is sufficient to bound the number of points used by any deterministic algorithm against the above input distribution.

We first bound the probability that the algorithm stores some point f_i. Call this event A. If we assume that the algorithm did not store any of the points $\{f_1, \dots, f_i\}$ it follows that the points in bucket B_{i+1} all have the same distance to the stored points. This implies that we can assume that the decision which points of bucket B_{i+1} will be kept is already fixed. The probability that f_i is one of these points is bounded by the hypergeometric distribution with population $32 \cdot N^{2/3}$, $N^{1/3}$ samples and 1 success in both population and sample: $\frac{\binom{32\cdot N^{2/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{32\cdot N^{2/3}}{N^{1/3}}}$. Then the probability that no f_i is stored for any of the $4 \cdot N^{1/3}$ buckets can be lower bounded by

$$1 - \mathbb{P}[A] \quad \geq \quad \left(1 - \frac{\binom{32\cdot N^{2/3}-1}{N^{1/3}-1} \cdot \binom{1}{1}}{\binom{32\cdot N^{2/3}}{N^{1/3}}}\right)^{4\cdot N^{1/3}} = \left(1 - \frac{N^{1/3}}{32 \cdot N^{2/3}}\right)^{4\cdot N^{1/3}} \geq 1 - \frac{1}{8} = \frac{7}{8}.$$

Now we bound the probability that the algorithm retains any point from bucket t upon submission, which we call event B. Again, conditioned on the event that A does not hold ($\overline{A}$), the buckets from which the algorithm stores at least one point are fixed. The probability that B_t is among the stored buckets again follows a hypergeometric distribution with population $4 \cdot N^{1/3}$, $N^{1/3}$ samples and 1 success in both population and sample. Therefore $\mathbb{P}[B|\overline{A}] = \frac{\binom{4\cdot N^{1/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{4\cdot N^{1/3}}{N^{1/3}}} = \frac{1}{4}$. Since one of the events A or B has to hold for the algorithm to output a solution with approximation factor greater than $\frac{1}{3}$, the probability that an algorithm storing less than $N^{1/3}$ points produces a solution with the desired approximation guarantee is at most $\mathbb{P}[A \cup B] \leq \mathbb{P}[A] + \mathbb{P}[B] = \mathbb{P}[A] + \mathbb{P}[B|A] \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \cdot \mathbb{P}[\overline{A}] \leq 2 \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \leq \frac{2}{8} + \frac{1}{4} = \frac{1}{2}$. ◀

We only briefly describe the k-center lower bound. The instance is also divided into sufficiently large buckets, from which the algorithm is forced to discard one point each. The main difference with the previous proof will be that the distances between all the points (except for a randomly chosen missing point f_t) are 2 and the distance from f_t to the more recent buckets is 4.

▶ **Theorem 12.** *For windows of size N, any randomized sliding window algorithm achieving an approximation factor less than 4 with probability greater than $\frac{1}{2}$ for the distance oracle metric 2 center problem with constant aspect ratio requires $\Omega(\sqrt[3]{N})$ points.*

References

1. Pankaj K. Agarwal, Jirí Matousek, and Subhash Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom.*, 1:189–201, 1991.
2. Pankaj K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72(1):83–98, 2015.
3. Brian Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 234–243, 2003.
4. Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering on sliding windows in polylogarithmic space. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 350–364, 2015.

5 Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1374–1390, 2016.

6 Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010.

7 Timothy M. Chan and Vinayak Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Comput. Geom.*, 47(2):240–247, 2014.

8 Timothy M. Chan and Bashir S. Sadjad. Geometric optimization problems over sliding windows. *Int. J. Comput. Geometry Appl.*, 16(2-3):145–158, 2006.

9 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 626–635, 1997.

10 Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 30–39, 2003.

11 Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 337–348, 2013.

12 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

13 Joan Feigenbaum, Sampath Kannan, and Jian Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41(1):25–41, 2004.

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

15 Sudipto Guha. Tight results for clustering and summarizing data streams. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 268–275, 2009.

16 Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33(3):533–550, 1986.

17 Piotr Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 539–545, 2003.

18 Sang-Sub Kim and Hee-Kap Ahn. An improved data stream algorithm for clustering. *Comput. Geom.*, 48(9):635–645, 2015.

19 Richard Matthew McCutchen and Samir Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 165–178, 2008.

20 Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

21 Hamid Zarrabi-Zadeh. Core-preserving algorithms. In *Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008*, 2008.

22 Hamid Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, 2011.