International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-022-00318-z

REGULAR PAPER

Adaptive k-center and diameter estimation in sliding windows

Paolo Pellizzoni[1] · Andrea Pietracaprina[1] · Geppino Pucci[1]

Received: 2 February 2021 / Accepted: 9 March 2022

Abstract
In this paper we present novel streaming algorithms for the k-center and the diameter estimation problems for general metric spaces under the sliding window model. The key idea behind our algorithms is to maintain a small coreset which, at any time, allows to compute a solution to the problem under consideration for the current window, whose quality can be made arbitrarily close to the one of the best solution attainable by running a polynomial-time sequential algorithm on the entire window. Remarkably, the size of our coresets is independent of the window length and can be upper bounded by a function of the target number of centers (for the k-center problem), of the desired accuracy, and of the characteristics of the current window, namely its doubling dimension and aspect ratio. One of the major strengths of our algorithms is that they adapt obliviously to these two latter characteristics. We also provide experimental evidence of the practical viability of the algorithms and their superiority over the current state of the art.

Keywords K-center · Diameter estimation · Data streams · Sliding window model · Coreset · Doubling-dimension · Approximation algorithms

1 Introduction

In several modern application domains (e.g., social networks, online finance, online transaction systems), data are generated in a continuous fashion, and at such a high rate that their processing requires on-the-fly computation which can afford to maintain only a small portion of the data in memory. This computational scenario is captured by the well-known streaming model, which has received ever-increasing attention in the literature over the last two decades [26,30,31,35,38]. In some prominent applications, it is also important that older data in the stream (i.e., those outside a sliding window containing the N most recent data items) be considered "stale" and thus be disregarded in the computation. As an example, consider the problem of detecting fraudulent credit card use, where it is essential to detect a change in the recent spending patterns. For this latter setting, an important variant of the streaming model, known as the *sliding window model*, was introduced in [16].

In this paper, we design, analyze, and experiment with novel streaming algorithms, under the sliding window model, for two key problems in data analysis, namely, the k-center and the diameter estimation problems. Let W be a set of points from a metric space. The k-center problem requires that, given a parameter $k < |W|$, a subset $C \subset W$ of k centers be identified minimizing the *radius* $r_C(W)$, defined as the maximum distance of any point of W from its closest center. This problem is a fundamental primitive in the realms of clustering and facility location, with important applications in a variety of domains such as database search, bioinformatics, pattern recognition, networking, operation research, and many more [4,25,40]. The diameter estimation problem requires to compute (or approximate) the maximum distance Δ_W between two points of W, which is a key indicator of the spread of the data.

Our algorithms improve over the state of the art in several directions, as described in Sect. 1.2.

1.1 Related work

In the standard static sequential setting, it is well known that for general metric space the k-center problem is NP-hard,

✉ Andrea Pietracaprina
andrea.pietracaprina@unipd.it

Paolo Pellizzoni
paolo.pellizzoni@studenti.unipd.it

Geppino Pucci
geppino.pucci@unipd.it

[1] Department of Information Engineering, University of Padova, Padova, Italy

admits a 2-approximation algorithm, and, for any $\epsilon > 0$, it is not $(2 - \epsilon)$-approximable unless P=NP [20].

The problem has also been studied in the fully dynamic setting where the input pointset changes dynamically through insertions of new points or deletions of existing points, and, at any time, the algorithm must be able to return an accurate solution for the current pointset in a time substantially smaller than the time required to compute the solution from scratch. In [13] the authors developed a $(2+\epsilon)$-approximation algorithm for the fully dynamic k-center problem on general metric spaces, with update time independent of the input size. For a given query point x, the algorithm can establish whether x is a center in constant time, and return the cluster of x (i.e., all points whose closest center is x) in time proportional to the cluster size. These results have been recently improved in [21] for spaces of constant doubling dimension using a navigating net data structure, and in [5] for general metric spaces using a reduction to the fully-dynamic maximal independent set problem. However, we remark that these fully dynamic algorithms store in memory a number of points linear with the size of the set of interest, as they rather target good query time/approximation tradeoffs, irrespective of the memory usage. For this reason they cannot be utilized in the sliding window model, where the size of the working memory, which is the premium resource to be optimized, must be substantially smaller than (and possibly independent of) the size of the set of interest.

In the standard streaming model, McCutchen and Khuller [33], and, independently, Guha [23], presented algorithms which maintain a $(2 + \epsilon)$-approximation to the k-center problem for the entire set of points processed from the beginning of the stream, using working memory polynomial in k and $1/\epsilon$. In the more restrictive sliding window model, which is the focus of this paper, Cohen-Addad et al. [14] presented an algorithm which is able to compute a $(6+\epsilon)$-approximation to the k-center problem for the current window, from only $O\left(k\epsilon^{-1}\log\alpha\right)$ points stored in the working memory, where α is the *aspect ratio* of the entire stream, that is, the ratio between the maximum and minimum distance between any two points of the stream. At any time, the algorithm requires $O\left(k\epsilon^{-1}\log\alpha\right)$ update time for handling the new point arrived from the stream, and $O\left(k^2\epsilon^{-1}\log\alpha\right)$ time to return the approximate solution for the current window. One of the practical limitations of this algorithm is that it assumes prior knowledge of the aspect ratio α, thus the algorithm is inapplicable for unknown or unbounded values of α. In the same paper, the authors also show that, for general metric spaces, any algorithm for the 2-center problem that achieves an approximation ratio less than 4 requires working memory of size $\Omega\left(N^{1/3}\right)$, where N is the window length. In a recent unpublished manuscript, Kim [28] improved the result in [14] for Euclidean spaces, by presenting an algorithm which attains a $(2+2\sqrt{3}+\epsilon)$-approximation through a coreset-based approach. The algorithm makes crucial use of specific properties of Euclidean spaces, hence, it is not immediately portable to general spaces. The author also claims that a $(2 + \epsilon)$-approximation is achievable for constant-dimensional Euclidean spaces, and that the algorithm can be made oblivious to the aspect ratio α. However, due to the missing details, it is not immediate to fully reconstruct these stated improvements.

For what concerns diameter estimation, it is shown in [23] that, in the streaming setting, in order to approximate the diameter within any factor strictly less than 2 for general metrics, a working memory at least proportional to the size of the stream is required. For Euclidean spaces of low dimension d, there is a streaming $(1+\epsilon)$-approximate algorithm requiring $O\left((1/\epsilon)^{(d-1)/2}\right)$ working memory [2], while for streams of higher dimensionality [3] presents an algorithm returning $(\sqrt{2} + \epsilon)$-approximate solutions using $O\left(d\epsilon^{-3}\log(1/\epsilon)\right)$ working memory, and [27] presents an algorithm which can approximate the diameter within a factor $c > \sqrt{2}$ using $O\left(dn^{1/(c^2-1)}\log n\right)$ working memory, where n is the size of the stream. A naive 2-approximation for the diameter of the entire stream is attainable in constant working memory by simply accumulating the maximum distance of all points in the stream from the first one. However, this approach cannot be effectively used in the sliding window model, because of the difficulty of maintaining the maximum distance from the first point of each window.

In [14], a streaming algorithm under the sliding window model is presented which, for any constant $\epsilon > 0$, is able to return a $(3+\epsilon)$-approximation to the diameter of the current window in general metric spaces using working memory of size $O\left(\log(\alpha)/\epsilon\right)$, where, again, α represents the aspect ratio of the entire stream and must be known in advance. In the same paper, the authors prove that, under reasonable assumptions, obtaining an approximation ratio less than 3 in general spaces requires $\Omega\left(N^{1/3}\right)$ working memory, where N is the length of the window. For Euclidean spaces of dimension d, a $(1 + \epsilon)$-approximation algorithm in the sliding window model is presented in [18], which uses a working memory of $O\left((1/\epsilon)^{(d+1)/2}\log^3 N(\log\alpha + \log\log N + (1/\epsilon))\right)$ bits. For constant d, the working memory requirement has been improved to $O\left((1/\epsilon)^{(d+1)/2}\log(\alpha/\epsilon)\right)$ [12].

For the smallest enclosing ball problem, which is closely related to the diameter estimation, [3] presents a streaming algorithm that can maintain a $((1+\sqrt{3})/2+\epsilon)$-approximate solution using $O\left(d\epsilon^{-3}\log(1/\epsilon)\right)$ working memory. In the more restrictive sliding window model, only a $(9.66 + \epsilon)$-approximation is known [42] which uses sub-polynomial working memory.

In the realm of large graph analytics, diameter approximation (under the shortest path metric) has been extensively addressed in the distributed setting (see [8,39] and references therein).

Another relevant variation on the diameter estimation problem is the computation of the α-effective diameter, which is defined as the α-th quantile of the distances between all pairs of elements. This concept was first introduced in [34] as a noise-robust alternative to diameter in the context of network analysis, but can be easily generalized to arbitrary metric spaces. Recently, both [37] and [17] independently presented sliding window algorithms for the effective diameter estimation problem in general metric spaces.

Finally, the sliding window model has recently been addressed in a large number of research papers, which provide algorithms for a wide variety of optimization problems, including k-means and k-medians [6], diversity maximization [7], k-center with outliers [17,37], submodular optimization [19], and heavy hitters [41].

1.2 Our contribution

In this work, we present streaming algorithms for the k-center and diameter estimation problems under the sliding window model. Our algorithms are coreset based, in the sense that they maintain a small subset of representative points embodying an accurate solution for the current window. More specifically, our algorithms rely on the data structures used in [14] to obtain an initial reasonable estimate of the optimal k-center radius, or the diameter, for the current window. These structures are paired with additional ones which leverage the initial estimate to maintain a coreset containing better representatives for the points of the current window. The working memory used by the algorithms is analyzed as a function of k, of a precision parameter ϵ, related to the desired approximation guarantee, and of the *doubling dimension* and of the aspect ratio of the current window. The doubling dimension, which is formally defined in Sect. 2, generalizes the notion of Euclidean dimensionality, and, as our results show, is related to the increasing difficulty of approximating the solution to the above problems when its value grows.

Consider a stream S of points from a metric space under the sliding window model. Let $\epsilon > 0$ denote a fixed, user-defined, precision parameter, and let α_W and D_W denote, respectively, the aspect ratio and the doubling dimension of the current window W. The two main theoretical results in this paper are the following:

- An algorithm that, at any time, is able to return a $(2+\epsilon)$-approximate solution to the k-center problem for the current window W, using working memory $M = O\left(k \log(\alpha_W)(c/\epsilon)^{D_W}\right)$, where $c > 1$ is a suitable constant, and α_W and D_W are, respectively, the aspect ratio and the doubling dimension of W. The update time required to handle each point is $O\left(M + k^2\right)$, while the query time to return the solution for the current window is $O\left(k^2(\log\log(\alpha_W) + (c/\epsilon)^{D_W})\right)$. (See Theorem 5 for detailed bounds.)
- An algorithm that, at any time, is able to return a lower estimate to the diameter Δ_W of the current window W, using working memory $M = O\left(\log(\alpha_W)(c/\epsilon)^{D_W}\right)$, where $c > 1$ is a suitable constant, and α_W and D_W are, respectively, the aspect ratio and the doubling dimension of W. The update time required to handle each point is $O(M)$, while the query time is $O\left((\log\log(\alpha_W) + (c/\epsilon)^{2D_W})\right)$, to ensure that the estimate is within a factor $(1+\epsilon)$ from Δ_W, and it is $O\left((\log\log(\alpha_W) + (c/\epsilon)^{D_W})\right)$, to ensure that the estimate is within a factor $(2+\epsilon)$ from Δ_W. (See Theorem 6 for detailed bounds.)

It is important to remark that our algorithms are *fully oblivious* to both the aspect ratio α_W and to the doubling dimension D_W, in the sense that these values are not used explicitly by the algorithms but they are only employed to analyze their space and time requirements. This is a crucial feature since, in practice, estimates for α_W and D_W would be very difficult to obtain. Moreover, as desirable in the sliding window model, the amount of working memory used by the algorithms is independent of the window length, and, for constant ϵ and D_W, they grow asymptotically only as a function of α_W and k (resp., only as a function of α_W), for the k-center (resp., diameter) problem.

The main improvements of our algorithms with respect to the state-of-the-art for general metric spaces [14] are:

- For the k-center problem, the approximation ratio drops from $6+\epsilon$ to $2+\epsilon$, with a moderate increase in the working memory and update/query time requirements for windows of low-dimension. Moreover, our result shows that the aforementioned lower bound on the working memory size, proved in [14], can be beaten when the doubling dimension of the stream is small. In general, the approximation ratio of our algorithm can be made arbitrarily close to 2, which, under the hypothesis P≠NP, is the best approximation attainable by any polynomial-time sequential algorithm when run on the entire window with unbounded memory,
- For the diameter estimation problem, the approximation ratio drops from $3+\epsilon$ to $1+\epsilon$, thus almost matching the exact estimation, with a moderate increase in the working memory and update/query time requirements for windows of low-dimension. (In fact, the query time can be improved by settling for a $(2+\epsilon)$-approximation.) Thus, our algorithm almost provides an exact estimation, whose computation would require time quadratic in the window length.

– Our algorithms, while oblivious to the doubling dimension of the window, afford a dimensionality-sensitive analysis which yields sharper resource-accuracy tradeoffs. They are also oblivious to the aspect ratio of the window, unlike the algorithms in [14] which require explicit knowledge of the aspect ratio of the entire stream. Note that this latter aspect ratio can be much larger than the largest aspect ratio of any window.

Finally, to gauge the practicality of our approach, we implemented our algorithms and the ones by [14], and compared their performance. The experiments provide clear evidence that, when endowed with similar amounts of working memory, on real-world datasets almost always our algorithms yield significantly better approximation with comparable update and query times.

1.3 Novelty with respect to conference version

A preliminary version of this work appeared in the Proceedings of the *7th IEEE International Conference on Data Science and Advanced Analytics*, (DSAA 2020) [36]. The novel contributions of this work with respect to the preliminary conference version are the following:

– strengthened analysis of the algorithms' space and time requirements, which now depend on the doubling dimension of the current window rather than on the doubling dimension of the entire stream;
– a new technique to make the algorithms oblivious to the aspect ratio of the metric;
– application of our clustering approach to the diameter estimation problem, improving upon the results of [14];
– substantially richer experimental analysis.

1.4 Organization of the paper

The rest of the paper is organized as follows. Section 2 defines the problems formally, and introduces a number of technical notions which will be used throughout the paper. Section 3.1 contains the description and the analysis of the algorithm for the k-center problem. In particular, Sects. 3.1 and 3.2 describe and analyze a simpler version of the algorithm assuming that the aspect ratio of the entire stream be known. Section 3.3 shows how to make the algorithm oblivious to the aspect ratio, and how to weaken the dependence from the aspect ratio of the entire stream to the one of current window. Section 4 describes and analyzes our algorithm for diameter estimation. Section 5 presents the experimental results. Finally, Sect. 6 offers some concluding remarks.

2 Preliminaries

Consider a pointset W from some metric space with distance function $\text{dist}(\cdot,\cdot)$. For any point $p \in W$ and any subset $C \subseteq W$ we use the notation

$$\text{dist}(p, C) = \min_{q\in C} \text{dist}(p, q),$$

and define the *radius of* C *with respect to* W as

$$r_C(W) = \max_{p\in W} \text{dist}(p, C).$$

For a positive integer $k < |W|$, the *k-center problem* requires to find a subset $C \subseteq W$ of size k which minimizes $r_C(W)$. Note that any subset $C \subseteq W$ of size k induces immediately a partition of W into k clusters by assigning each point to its closest center (with ties broken arbitrarily). We say that $r_C(W)$ is the radius of such a clustering, and define

$$\text{OPT}_{k,W} = \min_{C\subseteq W, |C|=k} r_C(W)$$

to denote the radius achieved by an optimal solution to the problem.

As recalled in the introduction, the well-known greedy sequential algorithm by Gonzalez [20] (dubbed GON in the rest of the paper), provides a 2-approximation to the k-center problem running in $O(|W|k)$ time. The following useful fact, proved in [10, Lemma 1], states that GON, when run on any subset T of the pointset W, returns a clustering whose radius cannot be much larger than the radius of an optimal clustering of the entire pointset.

Fact 1 *For any subset* $T \subseteq W$, *with* $|T| > k$, *let* C *be the output of* GON *when run on* T. *We have* $r_C(T) \le 2 \cdot \text{OPT}_{k,W}$.

We also define the *diameter* of a pointset W as $\Delta_W = \max_{p,q\in W} \text{dist}(p, q)$. The diameter can be computed exactly in quadratic time and, it is easy to argue that, for an arbitrary point $p \in W$,

$$\max_{q\in W} \text{dist}(p, q) \le \Delta_W \le 2 \max_{q\in W} \text{dist}(p, q),$$

whence

$$\text{OPT}_{1,W} \le \Delta_W \le 2\text{OPT}_{1,W}.$$

In the standard *streaming* framework [26,31] the computation is performed by a single processor with a small working memory, and the input is provided as a continuous, possibly unbounded, stream of objects (points, in our case), arriving one at each time step, which is usually too large to fit in the working memory. Under the *sliding window* model, at each time t, a solution to the problem of interest should be computable for the pointset W_t represented by the last N points arrived in the stream, where N, referred to as *window length*, is a predetermined value known to the algorithm. More formally, for each input point p, let $t(p)$ denote its arrival time. At any time t, we have that $W_t = \{p | 0 \leq t - t(p) < N\}$.[1] Since N can still be much larger than the working memory size, the challenging goal in this setting is to guarantee the quality of the solution while storing an amount of data substantially smaller than the window length.

Consider a stream S of points from a metric space with distance function $\text{dist}(\cdot, \cdot)$, and with a sliding window of length N. We define the *aspect ratio* α *of* S as the ratio between the maximum distance and the minimum distance of any two distinct points of S. Similarly, at any time t, we define the *aspect ratio* α_W *of the current window* W as the ratio between the maximum and the minimum distance of any two distinct points of W. These values will play an important role in our algorithms.

In this paper, we present streaming algorithms for the k-center problem and for diameter estimation, under the sliding window model. Our algorithms maintain information about a judiciously selected subset of points of the current window, from which, at any time t, a succinct *coreset* $T \subseteq W$ can be extracted, so that a solution to the problem under consideration can be efficiently computed by running a sequential (approximation) algorithm on T. The quality of a coreset T is regulated by a user-defined accuracy parameter $\epsilon > 0$, as captured by the following definition.

Definition 1 Given a pointset W and a value $\epsilon > 0$, a subset $T \subseteq W$ is an *ϵ-coreset for W* (w.r.t. the k-center problem) if $\max_{p \in W} \text{dist}(p, T) \leq \epsilon OPT_{k,W}$.

In other words, the property of an ϵ-coreset T of W is that each point in W is "close" enough to some point in T, where closeness is defined as a function of ϵ and $OPT_{k,W}$. Our algorithms for both the k-center problem and diameter estimation crucially rely on ϵ-coresets complying with the above definition. (For the diameter estimation, the ϵ-coresets employed will be w.r.t the 1-center problem.)

The time and space performance of our algorithms will be analyzed in terms of parameters k, N, α (or α_W), ϵ, and of the dimensionality of the points in the current window. Since we target the applicability of our algorithms to arbitrary metric spaces, we will make use of the following, general notion of dimensionality. Let W denote a set of points from a metric space. For any $x \in W$ and $r > 0$, let the *ball of radius r centered at x*, denoted as $B(x, r)$, be the subset of points of W at distance at most r from x. The *doubling dimension* of W is the smallest value D such that any ball $B(x, r)$, with $x \in W$, is contained in the union of at most 2^D balls of radius $r/2$ suitably centered at points of W. The following important fact, which we will use in the analysis, was proved in [24]:

Fact 2 *Let W be a set of points from a metric space and let $Y \subseteq W$ be such that any two distinct points $a, b \in Y$ have pairwise distance $\text{dist}(a, b) > r$. If W has doubling dimension D, then for every $R \geq r$ and any point $x \in W$, we have $|B(x, R) \cap Y| \leq (4R/r)^D$.*

A prominent feature of our algorithm is that *it adapts automatically to the doubling dimension D of the window*, in the sense that the algorithm does not require explicit knowledge of D, and provides best performances for small values of D. The characterization of datasets (or metric spaces) through their doubling dimension has been used in the literature in several contexts, including routing [29], clustering [1,10], nearest neighbor search [15], machine learning [22], and diversity maximization [9].

3 K-center problem

In this section, we present our $(2+\epsilon)$-approximation streaming algorithm for the k-center problem, under the sliding window model. The section is organized as follows. A first version of the algorithm, which assumes the knowledge of the aspect ratio α of the entire stream, is presented in Sect. 3.1 and analyzed in Sect. 3.2. Subsequently, Sect. 3.3 shows how to make the algorithm oblivious to α. Moreover, the dependence of the algorithm's space and time performance on the aspect ratio will be restricted to the one of current window, rather than to the one of the entire stream.

3.1 Algorithm

We consider the k-center problem for a target number k of centers, an input stream S, and a window length N. Let *minDist* and *maxDist* denote, respectively, the minimum and maximum distances between any two distinct points of S. To simplify the presentation of the algorithm, we assume that the values *minDist* and *maxDist*, hence the aspect ratio of S $\alpha = maxDist/minDist$ are known to the algorithm.

For each point p we define its *Time-To-Live* (TTL), denoted by $\text{TTL}(p)$, as $N - (t - t(p))$, where t is the current time. When p arrives ($t = t(p)$), its TTL is N, the window length, and, from that time on, $\text{TTL}(p)$ decreases of one unit at every new arrival. To avoid repeated updates of the TTL of points stored in the working memory, we assume that with each point p in the working memory we store the value $t(p)$, which allows to immediately compute its TTL, given the current time t and N. We say that a point p *expires* when it leaves the current window W, that is, when $\text{TTL}(p) = 0$. In

[1] For ease of notation, in what follows, we will omit the subscript in W_t, if clear from the context.

the analysis we will also consider points with negative TTL, that is, points that have expired at some previous time step.

For a user-defined constant $\beta > 0$, let

$$\Gamma = \{(1+\beta)^i : \lfloor \log_{1+\beta} minDist \rfloor \le i \le \lceil \log_{1+\beta} maxDist \rceil\},$$

and note that $|\Gamma| = O(\log(\alpha)/\log(1+\beta))$. As in [14], our algorithm runs several parallel instances, where each instance uses a different value $\gamma \in \Gamma$ as a guess of the optimal radius of a clustering of the current window. For each guess γ, the algorithm maintains two types of points belonging to the current window W: *validation points* (*v-points* for short) which enable to assess whether γ is a constant approximation to the optimal radius $\mathrm{OPT}_{k,W}$, and *coreset points* (*c-points* for short) which are those from which the coreset is extracted.

For each $\gamma \in \Gamma$, validation points are in turn organized into three (not necessarily disjoint) sets, namely AV_γ, RV_γ and OV_γ. Coreset points are similarly partitioned into sets A_γ, R_γ and O_γ. The sets of validation points serve the same purpose as those used in [14]. In broad terms, the set AV_γ (*attraction v-points*), whose size is upper bounded by $k+2$, contains centers of clusters of radius at most 2γ, which cover all points of W when γ is a valid guess for $\mathrm{OPT}_{k,W}$ (that is, $\mathrm{OPT}_{k,W} \le \gamma$). We say that a point p is *v-attracted* by $v \in AV_\gamma$ if $\mathrm{dist}(p, v) \le 2\gamma$. The set RV_γ (*representative v-points*) contains, for each $v \in AV_\gamma$, its *representative* $\mathrm{repV}_\gamma(v)$, defined as the newest point (that is, the point with the largest TTL) among those v-attracted by v. When v expires, its representative $\mathrm{repV}_\gamma(v)$ becomes an *orphan*, and it is moved to the set OV_γ (*orphan v-points*).

Let $\epsilon > 0$ be a user-defined precision parameter. The three sets of coreset points are used to refine the coverage provided by the validation points, so to make sure that, for valid guesses of γ, they contain an ϵ-coreset for the current window. Let $\delta = \epsilon/(1+\beta)$. The set A_γ (*attraction c-points*) contains centers that refine the clusters around the attraction v-points by reducing their radius by a factor $O(\delta)$. We say that a point p is *c-attracted* by $a \in A_\gamma$ if $\mathrm{dist}(p, a) \le \delta\gamma/2$. The sets R_γ and O_γ play, for c-points, the same role played by RV_γ and OV_γ for v-points. Thus, the set R_γ (*representative c-points*) contains a representative $\mathrm{repC}_\gamma(a)$ for each $a \in A_\gamma$, which is the newest point among those c-attracted by a. When a expires, its representative $\mathrm{repC}_\gamma(a)$ becomes an orphan and it is moved to the set O_γ (*orphan c-points*).

Observe that a point q can be a representative for several attraction v-points (resp., c-points). In that case, we assume that a distinct copy of q is maintained in RV_γ (resp., R_γ), one for each $v \in AV_\gamma$ (resp., $a \in A_\gamma$) such that $q = \mathrm{repV}_\gamma(v)$ (resp., $q = \mathrm{repC}_\gamma(a)$).

At every time step, a number of points, including those that expires at that step, are removed from the sets of validation and coreset points, so to keep their sizes under control. The interplay between validation and coreset points is the following. At any time t, the validation points enable to identify a suitable guess $\hat{\gamma}$ which is within a constant factor from the optimal value $\mathrm{OPT}_{k,W}$. Then, the set $R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ provides a good coreset from which an accurate final solution to k-center for W can be computed, using algorithm GON.

Our approach is described in detail by the following pseudocode, which consists of three procedures: UPDATE(p) describes the processing of each point p of the stream; INSERTVALIDATION(p, γ) is invoked inside UPDATE(p) when p must be added to AV_γ; finally, QUERY(), if invoked at time t, returns the coreset where algorithm GON can be run to return the solution.

```
UPDATE(p)
1  for each $\gamma \in \Gamma$
2      for each expired $v \in AV_\gamma$
3          $AV_\gamma = AV_\gamma \setminus \{v\}$
4          Move $\mathrm{repV}_\gamma(v)$ from $RV_\gamma$ to $OV_\gamma$
5      for each expired $a \in A_\gamma$
6          $A_\gamma = A_\gamma \setminus \{a\}$
7          Move $\mathrm{repC}_\gamma(v)$ from $R_\gamma$ to $O_\gamma$
8      Remove expired points from $OV_\gamma$ and $O_\gamma$
9      $EV = \{v \in AV_\gamma : \mathrm{dist}(p, v) \le 2\gamma\}$
10     $E = \{a \in A_\gamma : \mathrm{dist}(p, a) \le \delta\gamma/2\}$
11     if $EV == \emptyset$
12         INSERTVALIDATION($p, \gamma$)
13     else
14         for each $v \in EV$
15             set $\mathrm{repV}_\gamma(v) = p$ in $RV_\gamma$
16     if $E == \emptyset$
17         $A_\gamma = A_\gamma \cup \{p\}$
18         $\mathrm{repC}_\gamma(p) = p$
19         $R_\gamma = R_\gamma \cup \{\mathrm{repC}_\gamma(p)\}$
20     else
21         for each $a \in E$
22             set $\mathrm{repC}_\gamma(a) = p$ in $R_\gamma$
```

```
INSERTVALIDATION(p, γ)
1  $AV_\gamma = AV_\gamma \cup \{p\}$
2  $\mathrm{repV}_\gamma(p) = p$
3  $RV_\gamma = RV_\gamma \cup \{\mathrm{repV}_\gamma(p)\}$
4  if $|AV_\gamma| > k + 1$
5      $v_{old} = \mathrm{argmin}_{v \in AV_\gamma} \mathrm{TTL}(v)$
6      $AV_\gamma = AV_\gamma \setminus \{v_{old}\}$
7      Move $\mathrm{repV}_\gamma(v_{old})$ from $RV_\gamma$ to $OV_\gamma$
8  if $|AV_\gamma| > k$
9      $t_{\min} = \min_{v \in AV_\gamma} \mathrm{TTL}(v)$
10     for each $q \in A_\gamma$,
11         if $\mathrm{TTL}(q) < t_{\min}$
12             $A_\gamma = A_\gamma \setminus \{q\}$
13             Move $\mathrm{repC}_\gamma(q)$ from $R_\gamma$ to $O_\gamma$
14     Remove from $OV_\gamma$ and $O_\gamma$ all $q$ with $\mathrm{TTL}(q) < t_{\min}$
```

```
QUERY()
1  for increasing values of $\gamma \in \Gamma$ such that $|AV_\gamma| \le k$
2      $C = \emptyset$
3      for each $p \in AV_\gamma \cup OV_\gamma \cup RV_\gamma$
4          if $(C = \emptyset)$ or $(\mathrm{dist}(p, C) > 2\gamma)$
5              $C = C \cup \{p\}$
6      if $|C| \le k$
7          return $R_\gamma \cup O_\gamma$
```

We remark that values of $\delta \geq 4$ would be uninteresting since, in this case, the coreset points would not offer any refinement over the coverage provided by the validation points. Therefore, in what follows, we assume that ϵ and β are fixed so that $\delta \leq 4$.

3.2 Algorithm analysis

Suppose that Procedure UPDATE(p) is applied to every point p of the input stream S, upon arrival. In this section, we show that, at any time, Procedure QUERY (if invoked after UPDATE(p) has finished processing the latest point p arrived) returns an ϵ-coreset for the current window W, and that, by running GON on such a coreset, a $(2+\epsilon)$-approximate solution to the k-center problem for W is obtained. Moreover, we will analyze the amount of working memory the time required to process each point of the stream.

The following technical lemma states the main invariants maintained by Procedure UPDATE, which will be crucial for the analysis.

Lemma 1 *For every $\gamma \in \Gamma$, the following invariants hold at the end of each execution of Procedure* UPDATE(p)*, with respect to the window W containing p as its last point.*

1. *If $|AV_\gamma| \leq k$, then:*
   (a) $\max_{q\in W} \text{dist}(q, R_\gamma \cup O_\gamma) \leq \delta\gamma$;
   (b) $\max_{q\in W} \text{dist}(q, RV_\gamma \cup OV_\gamma) \leq 4\gamma$.
2. *If $|AV_\gamma| > k$, then:*
   (a) For every $q \in W$ with $\text{dist}(q, R_\gamma \cup O_\gamma) > \delta\gamma$*, then* $\text{TTL}(q) < \min_{v\in AV_\gamma} \text{TTL}(v)$.
   (b) For every $q \in W$ with $\text{dist}(q, RV_\gamma \cup OV_\gamma) > 4\gamma$*, then* $\text{TTL}(q) < \min_{v\in AV_\gamma} \text{TTL}(v)$.

Proof The proof for Invariants 1(b) and 2(b) follow the lines of the argument in [14], but we include it for completeness. For convenience, we subdivide the time in *steps*, where each step processes a point of the stream. It is easy to see that the invariants hold at the end of Step 0, which we consider as the beginning of the stream before the first point arrives. We suppose that the invariants hold at the end of Step $t-1$, for some $t > 0$, and show that they are maintained at the end of Step t. In the proof, we assume the following ordering of the activities of Step t: first, the point whose TTL goes to 0 expires and is thus excluded from the current window W; then, the new point p arrives and UPDATE(p) is executed; and, finally, at the end of UPDATE(p), p is included in the current window W. For each point $q \in W$ we define its *v-attractor* (resp., *c-attractor*) as the oldest attraction v-point (resp., attraction c-point) which was at distance at most 2γ (resp. $\leq \delta\gamma/2$) from q when q entered W. For simplicity, we define a number of *checkpoints* in the execution of Step t and show that if the invariants hold prior to each checkpoint, they also hold at the checkpoint. All the line numbers are, unless explicitly specified, referred to procedure UPDATE.

Checkpoint 1: the invariants hold after the point with TTL=0 expires. This is immediate to see, since we are only removing a point from the window, but the point is not yet removed from the sets stored in memory which it belongs to, if any.

Checkpoint 2: the invariants hold after Line 10. If $|AV_\gamma| \leq k$ before UPDATE(p) starts, it stays this way after Line 10, since Lines 1-10 do not add new points to AV_γ, thus we only need to prove that Invariant 1 is maintained. We will do the argument for 1(a), since the one for 1(b) is virtually identical. If the expired point is $o \in O_\gamma$, its removal in Line 8 does not affect the invariant. Indeed, if a point q violated 1(a) after the expiration of o, it would imply that o and q shared the same c-attractor, but, in this case, q would have expired before o and could not belong to W. If instead, the expired point is $a \in A_\gamma$ then its representative repC(a) is moved to O_γ. If a represents itself (i.e., $a = \text{repC}(a)$), then repC(a) will be also removed from the orphans and the considerations made above apply, otherwise the union $R_\gamma \cup O_\gamma$ remains unchanged. Note that no point of R_γ can expire unless its c-attractor also expires but, in this case, the point is moved to the orphan set, and this corresponds to the case considered above when $a = \text{repC}(a)$ expires. Consider now the case when $|AV_\gamma| > k$ before UPDATE(p) starts (note that it must necessarily be $|AV_\gamma| = k+1$). If a $v \in AV_\gamma$ expired, v is removed from AV_γ hence $|AV_\gamma| = k$ after Line 10, hence it suffices to prove that Invariant 1 holds. Note that all the points q such that $\text{dist}(q, R_\gamma \cup O_\gamma) > \delta\gamma$ already expired due to the fact that 2(a) holds at the beginning of UPDATE(p). Similarly, it can be argued that all points q such that $\text{dist}(q, RV_\gamma \cup OV_\gamma) > 4\gamma$ already expired. Consider now the case $|AV_\gamma| = k+1$ after Line 10, and let us first show that 2(a) holds. If a point q violates 2(a) this implies that a point $o \in O_\gamma$ with the same c-attractor as q has expired, but then q must have expired prior to o, hence it cannot belong to W. A similar argument can be used to prove 2(b).

Checkpoint 3: the invariants hold after Line 15. First, consider the case $EV = \emptyset$. Then INSERTVALIDATION(p, γ) is called to insert p in AV_γ. If, at the end of this call, we have $|AV_\gamma| \leq k$, then it was also $|AV_\gamma| \leq k$ at the start of the call. As a consequence, Invariant 1 must hold since, in this case, the call does not delete any point. Otherwise, at the end of INSERTVALIDATION(p, γ), it must be $|AV_\gamma| = k+1$ and we need to prove that Invariant 2 holds. Consider first 2(a). For any point q whose distance from $R_\gamma \cup O_\gamma$ becomes $> \delta\gamma$, there must be an orphan $o \in O_\gamma$ with the same c-attractor as q, which has been deleted in Line 14 of INSERTVALIDATION, hence $\text{TTL}(q) \leq \text{TTL}(o) < min_{v\in AV_\gamma} \text{TTL}(v)$. A symmetrical argument applies to prove 2(b). In case $EV \neq \emptyset$, we replace each representative repV(v) of $v \in EV$, with the new point p. Note that both $\mathrm{repV}(v)$ and p are *v-attracted* by the same point v, so, all points with the same v-attractor as $\mathrm{repV}(v)$ are contained in the 4γ-ball centered in p, which suffices to prove both Invariants 1 and 2.

Checkpoint 4: the invariants hold after Line 22. If $E = \emptyset$, then no point is deleted in Lines 16-22, thus the two invariants will hold. Otherwise, if $E \neq \emptyset$, we replace each representative c-point $\mathrm{repC}(a)$, $a \in E$ with the new point p. Since $\mathrm{repC}(a)$ and p are *c-attracted* by the same point a, all points with the same c-attractor as $\mathrm{repC}(a)$ are contained in the $\delta\gamma$-ball centered in p, which suffices to prove that Invariants 1 and 2 hold.

Checkpoint 5: the invariants hold after the new point p is inserted into the active window. If p has been inserted into AV_γ, then p has also been inserted into RV_γ, hence $\mathrm{dist}(p, RV_\gamma) = 0$. Else, there exists a point $v \in AV_\gamma$ such that $\mathrm{dist}(p, RV\gamma) \leq \mathrm{dist}(p, v) + \mathrm{dist}(v, RV_\gamma) \leq 4\gamma$. Similarly, it must hold that either $\mathrm{dist}(p, R\gamma) = 0$ (case $E = \emptyset$) or $\mathrm{dist}(p, R_\gamma) \leq \delta\gamma$ (case $E \neq \emptyset$), and the two invariants follow. □

Consider a time step t and let T be the set of points returned by Procedure QUERY invoked after the execution of UPDATE(p), where p is the t-th point of the stream. Let also W be the current window containing p as its last point. We have:

Lemma 2 *T is an ϵ-coreset for W w.r.t. the k-center problem.*

Proof It can be easily seen that for any guess γ such that either $|AV_\gamma| > k$, or $|AV_\gamma| \leq k$ and the set C computed by QUERY contains $> k$ points, there at least $k + 1$ points of W at pairwise distance $> 2\gamma$, which immediately implies that $\gamma < \mathrm{OPT}_{k,W}$. Moreover, since Γ contains a guess $\gamma \geq$ *maxDist* $\geq \mathrm{OPT}_{k,W}$, the procedure will always determine a minimum guess $\hat{\gamma}$ such that both $|AV_{\hat{\gamma}}| \leq k$ and $|C| \leq k$. Then, since the values in Γ form a geometric progression of common ratio $(1+\beta)$, we obtain that $\hat{\gamma}/(1+\beta) < \mathrm{OPT}_{k,W}$. Also, since $|AV_{\hat{\gamma}}| \leq k$, Invariant 1(a) of Lemma 1 ensures that

$$\max_{p\in W} \mathrm{dist}(p, T) < \delta\hat{\gamma} = \epsilon\hat{\gamma}/(1+\beta) < \epsilon \cdot \mathrm{OPT}_{k,W},$$

and the lemma follows. □

The next theorem establishes the approximation factor of our algorithm.

Theorem 1 *By running Algorithm* GON *on T we obtain a $(2+\epsilon)$-approximate solution for the k-center problem on W.*

Proof Let C^{alg} be the set of centers returned by GON when run on T. Since T is a subset of W, by Fact 1 we have that for each $q \in T$, $\mathrm{dist}(q, C^{\mathrm{alg}}) \leq 2\cdot\mathrm{OPT}_{k,W}$. Moreover, since T is an ϵ-coreset for W (Lemma 2) we have that for each $p \in W$ there is $q \in T$ such that $\mathrm{dist}(p, q) \leq \epsilon \cdot \mathrm{OPT}_{k,W}$. By combining these two observations and applying the triangle inequality, we conclude that for each $p \in W$ we have $\mathrm{dist}(p, C^{\mathrm{alg}}) \leq (2+\epsilon)\cdot\mathrm{OPT}_{k,W}$. □

The next two theorems establish the space and time requirements of our algorithm.

Theorem 2 *At any time t during the processing of the stream S, the sets stored in the working memory (i.e., AV_γ, RV_γ, OV_γ A_γ, R_γ, and O_γ, for every guess γ) contain*

$$O\left(k \cdot \frac{\log(\alpha)}{\log(1+\beta)}\left(\frac{32(1+\beta)}{\epsilon}\right)^{D_W}\right)$$

points, overall, where D_W is the doubling dimension of the current window W.

Proof Consider an arbitrary time t. Since $|\Gamma| = O(\log(\alpha)/\log(1+\beta))$, it is sufficient to show that for every $\gamma \in \Gamma$ the aggregate size of the sets of validation and coreset points is $O\left(k \cdot (32(1+\beta)/\epsilon)^{D_W}\right)$. We first show that $|AV_\gamma| \leq k+1$, $|RV_\gamma| \leq k+1$, $|OV_\gamma| \leq k+1$. The proof argument is the same as the one used in [14], but we report it for completeness. The bound on $|AV_\gamma|$ is explicitly enforced by INSERTVALIDATION which removes a point from AV_γ as soon as its size exceeds $k+1$. The bound on RV_γ follows from the fact that the algorithm makes sure that RV_γ contains exactly one representative for each $v \in AV_\gamma$. Indeed, when a point is removed from AV_γ, its representative is moved to OV_γ.

For what concerns the bound on $|OV_\gamma|$, let $v_1, v_2, \ldots$ be an enumeration of the points inserted in AV_γ at any time during the algorithm, ordered by arrival time. We now show that for every $i \geq 1$ we have $\mathrm{TTL}(v_{i+k+1}) > \mathrm{TTL}(\mathrm{repV}_\gamma(v_i)) \geq \mathrm{TTL}(v_i)$. Consider two cases. If v_i expires before v_{i+k+1} enters the window, then $\mathrm{TTL}(v_{i+k+1}) > \mathrm{TTL}(\mathrm{repV}_\gamma(v_i))$ because $\mathrm{repV}_\gamma(v_i)$ must have entered the window before v_i expired. Otherwise, upon insertion of v_{i+k+1} in AV_γ, there are $k+1$ points in AV_γ, so the algorithm deletes v_i as it is the oldest point in AV_γ. Then, again $\mathrm{TTL}(v_{i+k+1}) > \mathrm{TTL}(\mathrm{repV}_\gamma(v_i))$ because $\mathrm{repV}_\gamma(v_i)$ must have entered the window before v_i is deleted. At time t, let v_j be the last point that was removed from AV_γ, either because expired or deleted. By the property proved above, any point which has been representative of $v_{j-(k+1)}$ has a TTL smaller than $\mathrm{TTL}(v_j)$, thus it cannot be in memory at time t because it either expired or has been deleted by Line 14 of INSERTVALIDATION. This shows that $|OV_\gamma| \leq k+1$.

Next, we show that $|A_\gamma \cup R_\gamma \cup O_\gamma| \leq 6(k+1)(32/\delta)^{D_W}$, where D_W is the doubling dimension of the current window W. From the proof above we know that there are at most $k+1$ points in each of the sets AV_γ, RV_γ and OV_γ. By construction, we also know that the distance between any two points of A_γ is $\geq \delta\gamma/2$. We show that the points of A_γ are enclosed in at most $2(k+1)$ balls of radius 4γ. Consider two cases. If $|AV_\gamma| \leq k$, by Invariant 1(b) we have $\max_{q\in W} dist(q, RV_\gamma \cup OV_\gamma) \leq 4\gamma$, hence each $q \in A_\gamma$ is within one of the at most $2(k+1)$ balls of radius 4γ centered at the points of $RV_\gamma \cup OV_\gamma$. Instead, if $|AV_\gamma| = k+1$, then by Invariant 2(b) we have that for each $q \in W$ with $\text{TTL}(q) \geq \min_{v\in AV_\gamma} \text{TTL}(v)$ it holds that $\max_{q\in W} dist(q, RV_\gamma \cup OV_\gamma) \leq 4\gamma$. Recall that after we insert a new point in AV_γ, if the size exceeds k we delete from $|A_\gamma \cup R_\gamma \cup O_\gamma|$ all the points with TTL smaller than the smallest TTL of a point in AV_γ. Then after each execution of the procedure UPDATE, if $|AV_\gamma| = k+1$, each point in A_γ has TTL greater than the oldest point in AV. Thus, each $q \in A_\gamma$ is within a ball of radius 4γ from some point in $RV_\gamma \cup OV_\gamma$. By Fact 2, in each of these $2(k+1)$ balls, there can be at most $(32/\delta)^{D_W}$ points of A_γ, so $|A_\gamma| \leq 2(k+1)(32/\delta)^{D_W}$. Moreover, at any given time $|R_\gamma| = |A_\gamma|$, since the algorithm makes sure that R_γ contains exactly one representative for each $a \in A_\gamma$.

Let $k' = 2(k+1)(32/\delta)^{D_W}$ be the above upper bound on the size of A_γ. We are left to show that $|O_\gamma| \leq k'$. Let $a_1, a_2, \ldots$ be an enumeration of the points inserted in A_γ at any time during the algorithm, ordered by arrival time. We now show that for every $i \geq 1$ we have $\text{TTL}(a_{i+k'+1}) > \text{TTL}(\text{repC}_\gamma(a_i)) \geq \text{TTL}(a_i)$. It must hold that a_i expires or gets deleted before $a_{i+k'+1}$ enters the window, or otherwise, upon insertion of the new point $a_{i+k'+1}$, there would be $k'+1$ points in A_γ, which is impossible since k' is an upper bound to $|A_\gamma|$. Then, $\text{TTL}(a_{i+k'+1}) > \text{TTL}(\text{repC}_\gamma(a_i))$ because $\text{repC}_\gamma(a_i)$ must have entered the window before a_i expired or got deleted, which means that $\text{repC}_\gamma(a_i)$ must have entered the window before $a_{i+k'+1}$ enters the window. Let a_j be the last point that was removed from A_γ, either because expired or deleted. By the property proved above, any point which has been representative of $a_{j-(k'+1)}$ has a TTL smaller than $\text{TTL}(v_j)$, thus it cannot be in memory at time t because it either expired or has been deleted by Line 16 of INSERTVALIDATION. This shows that there can be at most k' points in O_γ. □

Theorem 3 *Procedure* UPDATE(p) *runs in time*

$$O\left(k \cdot \frac{\log(\alpha)}{\log(1+\beta)} \left(\frac{32(1+\beta)}{\epsilon}\right)^{D_W}\right),$$

while Procedure QUERY() *runs in time*

$$O\left(k^2 \cdot \frac{\log(\alpha)}{\log(1+\beta)} + k \cdot \left(\frac{32(1+\beta)}{\epsilon}\right)^{D_W}\right),$$

where D_W is the doubling dimension of the current window W.

Proof The time complexity of UPDATE(p) is dominated by the construction of the sets EV and E for each γ (Lines 9 and 10), which requires time linear in $|AV_\gamma| + |A_\gamma|$. The claimed bound follows by Theorem 2. For what concerns QUERY, we observe that, as shown in the proof of Theorem 2

$$|AV_\gamma| + |RV_\gamma| + |OV_\gamma| = O(k),$$

hence the identification of the minimum guess $\hat{\gamma}$ such that both $|AV_{\hat{\gamma}}| \leq k$ and $|C| \leq k$ can be easily accomplished in $O\left(k^2 \log(\alpha)/\log(1+\beta)\right)$ time. Finally, once $\hat{\gamma}$ has been found, returning $R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ takes time proportional to their size, which is $O\left(k(32(1+\beta)/\epsilon)^{D_W}\right)$ as argued in the proof of Theorem 2. □

We remark that the first term in the running time of Procedure QUERY() can be improved to $O\left(k^2 \log(\log(\alpha)/\log(1+\beta))\right)$, by using binary search over the values of Γ.

The following theorem summarizes the main features of our algorithm.

Theorem 4 *Consider a stream S of points from a metric space under the sliding window model. Let α be the aspect ratio of S, and let D_W be the doubling dimension of the current window W. For fixed parameters $\epsilon, \beta > 0$, at any time t our algorithm for the k-center problem requires working memory of size $M = O(k \cdot (\log(\alpha)/\log(1+\beta))$ $(32(1+\beta)/\epsilon)^{D_W})$, processes each point $p \in S$ in time $O(M)$, and is able to return a $(2+\epsilon)$-approximation to the k-center problem for W in time $O\left(k^2 \cdot \left(\log(\log(\alpha)/\log(1+\beta)) + (32(1+\beta)/\epsilon)^{D_W}\right)\right)$.*

Proof The theorem is an immediate consequence of Theorems 1, 2 and 3, and of the observation that computing the final solution on the coreset of size $O\left(k(32(1+\beta)/\epsilon)^{D_W}\right)$, returned by QUERY requires time $O\left(k^2(32(1+\beta)/\epsilon)^{D_W}\right)$. □

We remark that the amount of working memory required by the algorithm to analyze the entire stream S will depend on the maximum value of D_W, which is upper bounded by the doubling dimension of the entire stream but can in fact be substantially smaller.

3.3 Obliviousness to α

For convenience, the algorithm presented in Sect. 3.1 assumed explicit knowledge of the aspect ratio α of the entire stream S. In this section, we show how to make the algorithm oblivious to the aspect ratio, while keeping the same approximation quality, time and space requirements. In fact, the analysis will also show that the dependence on α of the time and space requirements of the oblivious algorithm can be weakened into a dependence on the value α_W the aspect ratio restricted to the current window W.

We observe that the knowledge of α required by the algorithm presented in Sect. 3.1 serves solely the purpose of identifying a spectrum of feasible values for the optimal cluster radius. This is reflected in the definition of Γ, which contains a geometric progression of values spanning the interval between the minimum and maximum distance of any two points of the stream. In the oblivious version, at any time t the algorithm maintains an analogous set Γ_t, spanning an interval delimited by a lower bound and an upper bound to the radius of the optimal clustering for the current window W.

Let $p_1, p_2, \ldots$ be an enumeration of all points of the stream S based on the arrival order. At every time $t > k$, let r_t be the minimum pairwise distance between the last $k+1$ points of the stream $(p_{t-k}, \ldots, p_{t-1}, p_t)$. It is easy to see that, for the current window W, $\mathrm{OPT}_{k,W} \geq r_t/2$. We require that, together with the other structures, the algorithm stores the last $k+1$ points arrived and maintains the value r_t, which can be computed with an extra $O\left(k^2\right)$ operations per step.

We also require that the algorithm maintains a tight upper bound on the diameter Δ_W of the current window W. More precisely, we require that the algorithm maintains, at any time t, a value M_t such that $M_t \geq \Delta_W$, with $M_t = \Theta\left(\Delta_W\right)$. To ease presentation, let us assume for now that M_t is available. At the end of this subsection, we will show how to augment the algorithm so that M_t can indeed be efficiently computed at each step.

Let the values β, ϵ and $\delta = \epsilon/(1+\beta)$ be defined as in Secti. 3.1, and recall that we assumed that $\delta \leq 4$ since larger values of δ are uninteresting for our algorithm. We define

$$\Gamma_t = \{(1+\beta)^i : \lfloor \log_{1+\beta} r_t/2 \rfloor \leq i \leq \lceil \log_{1+\beta} 2M_t/\delta \rceil\},$$

and observe that the definition of Γ_t is independent of α. The structure of the α-oblivious algorithm is identical to the one presented in the previous subsections, except that at each time t, the set of guesses Γ_t substitutes the fixed set of guesses Γ. The following claim shows that at any step t, it is sufficient that the algorithm maintains structures for guesses γ belonging to Γ_t.

Claim Consider the non-oblivious algorithm presented in Sect. 3.1. At any time t, Procedure QUERY() would be correct if the sets AV_γ, RV_γ, OV_γ, A_γ, R_γ, and O_γ satisfying the invariants stated in Lemma 1 were available only for $\gamma \in \Gamma_t$.

Proof We show that, even if available, the sets of validation and coreset points for values of γ outside Γ_t would never provide the final coreset returned by QUERY(). Consider a value $\gamma < \min \Gamma_t \leq r_t/2$, and observe that the last $k+1$ points of the stream, namely $p_{t-k}, \ldots, p_{t-1}, p_t$, are all at pairwise distance at least $r_t > 2\gamma$. Therefore, the non-oblivious algorithm would insert all of these points in AV_γ, hence $|AV_\gamma| = k+1$ and the value γ would not be considered in the outer loop of QUERY(). Let $\gamma_{\max} = \max \Gamma_t \geq 2M_t/\delta$. We show that at time t, when QUERY() considers $\gamma_{\max}$ it must find $|AV_{\gamma_{\max}}| \leq k$ and a set C of size at most k, since otherwise there would exist $k+1$ points at distance $> 2\gamma_{\max} \geq 4M_t/\delta \geq M_t$, which is impossible since M_t is an upper bound to the diameter Δ_W, where W is the current window up to point p_t. Thus, Procedure QUERY() surely returns a coreset for a value $\gamma \leq \gamma_{\max}$. □

We now show how to modify the algorithm so that, without the knowledge of α, at the end of each step t it is able to maintain the sets AV_γ, RV_γ, OV_γ, A_γ, R_γ, and O_γ satisfying the invariants stated in Lemma 1, limited to the guesses $\gamma \in \Gamma_t$. Suppose that this is the case for up to some time $t-1 > k$ and consider the arrival of p_t. First, the algorithm computes the new values r_t (as described above) and M_t (to be described later), and removes all sets relative to values of $\gamma \in \Gamma_{t-1} - \Gamma_t$. Then, if $r_t < r_{t-1}$, for each $\gamma \in \Gamma_t$ with $\gamma < \min \Gamma_{t-1}$, the algorithm sets $AV_\gamma = \{p_{t-k-1}, \ldots, p_{t-1}\} = RV_\gamma = A_\gamma = R_\gamma$ and $OV_\gamma = O_\gamma = \emptyset$. It is easy to argue that, for these values of γ, these sets satisfy the invariants of Lemma 1 at the end of step $t-1$. Also, if $M_t > M_{t-1}$, then for each $\gamma \in \Gamma_t$ with $\gamma > \max \Gamma_{t-1}$, the algorithm sets $AV_\gamma = \{p_{t-1}\} = RV_\gamma = A_\gamma = R_\gamma$ and $OV_\gamma = O_\gamma = \emptyset$. Again, for these values of γ, the sets satisfy the invariants of Lemma 1 at the end of step $t-1$, since for every point q in the window at that time we have $\mathrm{dist}(q, RV_\gamma) = \mathrm{dist}(q, R_\gamma) = \mathrm{dist}(q, p_{t-1}) \leq M_{t-1} \leq \delta\gamma/2 \leq 2\gamma \leq 4\gamma$, since we are assuming $\delta \leq 4$.

At this point, for every $\gamma \in \Gamma_t$ the algorithm has available sets AV_γ, RV_γ, OV_γ, A_γ, R_γ, and O_γ which satisfy the invariants of Lemma 1 at the end of step $t-1$. Then, it is sufficient to run UPDATE(p_t) to complete step t.

Computation of M_t It remains to show how to compute M_t at each step. To this purpose, we require that, for every $\gamma \in \Gamma_t$, a separate concurrent process to the main algorithm described above maintains a set of validation points for $k = 1$, which we refer to as $AV_\gamma^{(1)}$, $RV_\gamma^{(1)}$, $OV_\gamma^{(1)}$. Clearly, for each step $t \leq k+1$, maintaining these three sets and an upper bound M_t to Δ_W is trivial. Suppose now that these sets are available at the end of some step $t-1 > k$, for every $\gamma \in \Gamma_{t-1}$. and consider the arrival of p_t. We operate as follows. We first compute r_t. Then, for each $\gamma = (1+\beta)^i$, with $i \geq \lfloor \log_{1+\beta} r_t/2 \rfloor$, we perform the following sequence of steps until a given value $\hat{\gamma}$ is determined:

1. if $\gamma < \min \Gamma_{t-1}$ or $\gamma > \max \Gamma_{t-1}$, we generate $AV_\gamma^{(1)}$, $RV_\gamma^{(1)}$, and $OV_\gamma^{(1)}$ as explained above for the case of arbitrary k;
2. we update $AV_\gamma^{(1)}$, $RV_\gamma^{(1)}$, and $OV_\gamma^{(1)}$ to account for the arrival of p_t, performing the same operations that would be performed by UPDATE(p_t) with $k = 1$;

3. if $|AV_\gamma^{(1)}| = 1$ and $\max_{p\in OV_\gamma^{(1)}} \mathrm{dist}(p, a) \le 2\gamma$ where a is the only point in $AV_\gamma^{(1)}$, set $\hat\gamma = \gamma$ and exit.

Observe that this procedure is akin to the search for a feasible value of γ performed by Procedure QUERY, and will surely stop at a value $\hat\gamma = O(\Delta_W)$. Once $\hat\gamma$ is computed, M_t is set equal to $12\hat\gamma$, which, together with r_t computed before, determines the set Γ_t. At this point, Steps 1) and 2) above are repeated for all values of $\gamma \in \Gamma_t$, with $\gamma > \hat\gamma$, so to have the sets $AV_\gamma^{(1)}$, $RV_\gamma^{(1)}$, $OV_\gamma^{(1)}$ updated at the end of time t for every $\gamma \in \Gamma_t$.

The following lemma shows that M_t is indeed a tight upper bound on Δ_W.

Lemma 3 *We have:*

$$\Delta_W \le M_t < 6(1+\beta)\Delta_W,$$

where W is the current window, up to and including p_t.

Proof Let $AV_{\hat\gamma}^{(1)} = \{a\}$. The following sequence of inequalities follows by Invariant 1(b) of Lemma 1, by the properties of $RV_{\hat\gamma}^{(1)}$, and by the above definition of $\hat\gamma$.

$$\begin{aligned}\mathrm{OPT}_{1,W} &\le \max_{q\in W} \mathrm{dist}(p, a)\\ &\le \max_{q\in W} \mathrm{dist}(p, RV_{\hat\gamma}^{(1)} \cup OV_{\hat\gamma}^{(1)}) +\\ &\quad + \max_{p\in RV_{\hat\gamma}^{(1)} \cup OV_{\hat\gamma}^{(1)}} \mathrm{dist}(q, a))\\ &\le 4\hat\gamma + 2\hat\gamma = 6\hat\gamma.\end{aligned}$$

Therefore, it follows that $M_t = 12\hat\gamma \ge 2\mathrm{OPT}_{1,W} \ge \Delta_W$. For what concerns the upper bound on M_t, observe that either $\hat\gamma$ is the initial value in Γ_t, hence, $\hat\gamma \le r_t/2 \le \Delta_W/2$, or its preceding value in Γ_t, $\gamma' = \hat\gamma/(1+\beta)$, is such that $|AV_{\gamma'}^{(1)}| > 1$ or $AV_{\gamma'}^{(1)}$ contains only one point a and there exists a point $q \in OV_{\gamma'}^{(1)}$ with $\mathrm{dist}(q, a) > 2\gamma'$. In either case, we have that $\Delta_W > 2\gamma' = 2\hat\gamma/(1+\beta)$, which implies that $M_t < 6(1+\beta)\Delta_W$. □

The following theorem summarizes the main result regarding the α-oblivious version of our algorithm presented in this subsection.

Theorem 5 *Consider a stream S of points from a metric space under the sliding window model. Let α_W and D_W be, respectively, the aspect ratio and the doubling dimension of the current window W. For fixed parameters $\epsilon, \beta > 0$, at any time t the α-oblivious version of our algorithm for the k-center problem requires working memory of size $M = O\left(k \cdot (\log(\alpha_W)/\log(1+\beta))(32(1+\beta)/\epsilon)^{D_W}\right)$, processes each point $p \in S$ in time $O\left(M + k^2\right)$, and is able to return a $(2+\epsilon)$-approximation to the k-center problem for W in time $O\left(k^2 \cdot (\log(\log(\alpha_W)/\log(1+\beta)) + (32(1+\beta)/\epsilon)^{D_W})\right)$.*

Proof First we observe that $\Gamma_t \in O(\log(\alpha_W)/\log(1+\beta))$. Reasoning as in the proof of Theorem 2, it is immediate to see that, for each guess γ, the aggregate size of all sets of validation and coreset employed by the algorithm, including those required to compute M_t, is still $O\left(k(32(1+\beta)/\epsilon)^{D_W}\right)$. Hence the claimed bound on M follows. For what concerns the time required to process each point, the computation of r_t requires $O\left(k^2\right)$ time while the computation of M_t and the update of all sets maintained in the working memory requires time $O(M)$. Finally, the query time and the approximation factor follow from Theorem 4 substituting α with α_W. □

Similarly to what we remarked at the end of the previous subsection for the doubling dimension, we observe that now the amount of working memory required by the algorithm to analyze the entire stream S will depend on the maximum value of α_W, which is upper bounded by the aspect ratio α of the entire stream but can in fact be substantially smaller.

4 Diameter estimation

In this section, we present an algorithm for accurate diameter estimation in the sliding window model. Let S be a stream of points from a metric space, with a sliding window of length N. Fix an accuracy parameter $\epsilon > 0$ and let ϵ' be such that $\epsilon = \epsilon'/(1-\epsilon')$. Suppose that we run the α-oblivious algorithm for the k-center algorithm discussed in the previous section, with $k = 1$ and accuracy ϵ'. Also, in the algorithm, set $\delta = \epsilon'/(2(1+\beta))$ instead of $\delta = \epsilon'/(1+\beta)$. For a time step t, let T be the set of points returned by Procedure QUERY() after the t-th point has been processed with Procedure UPDATE. We have:

Lemma 4 *Let Δ_W and Δ_T be the diameters of the current window W and of the coreset T, respectively. Then*

$$\Delta_T \le \Delta_W \le (1+\epsilon)\Delta_T.$$

Proof The lower bound on Δ_W is immediate since $T \subseteq W$. For what concerns the upper bound, let $\hat\gamma$ be the smallest estimate for which both $|AV_{\hat\gamma}| \le 1$ and $|C| \le 1$ in Procedure QUERY(). By reasoning as in the proof of Lemma 2, we obtain that $\hat\gamma/(1+\beta) \le \mathrm{OPT}_{1,W}$. Hence,

$$\max_{p\in W} \mathrm{dist}(p, T) < \delta\hat\gamma = \frac{\epsilon'}{2(1+\beta)}\hat\gamma < \frac{\epsilon'}{2}\mathrm{OPT}_{1,W} \le \frac{\epsilon'}{2}\Delta_W.$$

For each point $p \in W$, we define its *proxy* point $\pi(p)$ as the closest point to p in T, namely, $\pi(p) = \mathrm{argmin}_{q\in T} \mathrm{dist}(p, q)$. Let p, q be two points of W such that $\mathrm{dist}(p, q) = \Delta_W$. We have that

$$\begin{aligned}\Delta_W &= \mathrm{dist}(p,q)\\ &\le \mathrm{dist}(p,\pi(p)) + \mathrm{dist}(\pi(p),\pi(q)) + \mathrm{dist}(q,\pi(q))\\ &\le \epsilon'\Delta_W + \mathrm{dist}(\pi(p),\pi(q))\\ &\le \epsilon'\Delta_W + \Delta_T.\end{aligned}$$

Therefore, $\Delta_T \ge (1-\epsilon')\Delta_W$ and, since $\epsilon = \epsilon'/(1-\epsilon')$, this implies $\Delta_W \le 1/(1-\epsilon')\Delta_T = (1+\epsilon)\Delta_T$. □

Let D_W be the doubling dimension of the current window. By repeating the argument in the proof of Theorem 2, we obtain that the coreset size is

$$\begin{aligned}|T| &= O\left(\left(\frac{32}{\delta}\right)^{D_W}\right)\\ &= O\left(\left(\frac{64(1+\beta)(1+\epsilon)}{\epsilon}\right)^{D_W}\right),\end{aligned}$$

since $\delta = \epsilon'/(2(1+\beta))$ and $\epsilon' = \epsilon/(1+\epsilon)$. Lemma 4 implies that an estimate of Δ_W within a factor $(1+\epsilon)$ can be obtained by computing the exact diameter of the coreset T in time $O\left(|T|^2\right)$. If, due to the exponential dependency on D_W, the coreset size is too large to afford a quadratic computation of its diameter, one can compute instead $\max_{q\in T}\mathrm{dist}(p,q)$, for an arbitrary point $p \in T$, which yields an estimate of Δ_T within a factor 2, whence and estimate of Δ_W within a factor $2(1+\epsilon)$, in linear time.

The following theorem summarizes the results of this section:

Theorem 6 *Consider a stream S of points from a metric space under the sliding window model. Let α_W and D_W be, respectively, the aspect ratio and the doubling dimension of the current window W. For fixed parameters $\epsilon, \beta > 0$, at any time t our algorithm for diameter estimation requires working memory of size $M = O\left((\log(\alpha_W)/\log(1+\beta))(64(1+\beta)(1+\epsilon)/\epsilon)^{D_W}\right)$, processes each point $p \in S$ (i.e. slides the window) in time $O(M)$, and it is able to return*

- *an estimate within a factor $(1+\epsilon)$ to the diameter Δ_W in time $O\left(\log\left(\frac{\log(\alpha_W)}{\log(1+\beta)}\right) + (64(1+\beta)(1+\epsilon)/\epsilon)^{2D_W}\right)$;*
- *an estimate within a factor $2(1+\epsilon)$ to the diameter Δ_W in time $O\left(\log\left(\frac{\log(\alpha_W)}{\log(1+\beta)}\right) + (64(1+\beta)(1+\epsilon)/\epsilon)^{D_W}\right)$.*

Proof The proofs of the bounds on M and on the processing time are obtained along the same lines as Theorem 4, by using $k = 1$, $\delta = \epsilon'/(2(1+\beta))$, with $\epsilon' = \epsilon/(1+\epsilon)$, and by substituting α with α_W, in the light of the result of Sect. 3.3. The bounds on the accuracy of the estimations and on the time required to compute them follow from Lemma 4 and the subsequent discussion on computing/estimating the diameter of the coreset, and from the fact that, for $k = 1$, Procedure QUERY() requires $O\left(\log(\log(\alpha_W)/\log(1+\beta)) + (64(1+\beta)(1+\epsilon)/\epsilon)^{D_W}\right)$ time. □

5 Experiments

To assess the practical viability of our approach, we designed a set of experiments to

- compare approximation quality, execution time, and memory usage (in terms of the number of points stored in the working memory) of our k-center algorithm against the state-of-the-art algorithm in [14];
- test the ability of our k-center algorithm to adapt, at each time t, to the specificity of the window W_t, as captured by its doubling dimension D_W and its aspect ratio α_W;
- compare approximation ratio, execution time, and memory usage of our $(1+\epsilon)$-approximate diameter algorithm against the state-of-the-art algorithm in [14].

5.1 Experimental testbed and datasets

All tests were executed using a Java 13 implementation of the algorithms on a Windows machine running on an AMD FX8320 processor with 12GB of RAM, and the running times of the procedures were measured with `System.nanoTime`. The points of the datasets are fed to the algorithms through the file input stream.

We experimented with datasets often used in the clustering literature. Specifically, we used both a low-dimensional dataset, `Higgs`, and for a higher-dimensional dataset, `Covertype`, which serves as a stress test for our dimensionality-sensitive approach. The `Higgs` dataset[2] contains 11 million points representing high-energy particle features generated through Monte-Carlo simulations. The points of this dataset have 28 attributes, 7 of which are a function of all the others. We considered only these seven "high-level" features, hence regard the data as points in $\mathbb{R}^7$ using Euclidean distance. The `Covertype` dataset[3] contains $\simeq 500$ thousand 54-dimensional points from geological observations of US forest biomes. We interpret the data as points in $\mathbb{R}^{54}$ using Euclidean distance. In some experiments, we will also use artificial datasets consisting points randomly extracted from (subspaces of) $\mathbb{R}^{100}$, using again Euclidean distance.

[2] http://archive.ics.uci.edu/ml/datasets/HIGGS

[3] https://archive.ics.uci.edu/ml/datasets/covertype

5.2 Comparison with the state-of-the-art k-center algorithm

We designed implementations of the non-oblivious and of the α-oblivious version of our k-center strategy, respectively referred to as OUR- SLIDING and OUR- OBLIVIOUS hereinafter, and an implementation of the state-of-the art sliding window algorithm in [14], referred to as CSS- SLIDING. Due to the NP-hardness of the k-center problem, it is unfeasible to compute the optimal solution for each window so to measure the exact approximation factor of the solutions returned by the algorithms. As a workaround, we assess the quality loss incurred by our space-restricted streaming algorithms with respect to the most accurate, polynomial-time sequential approximation GON running on the entire window, hence using unrestricted space.

For the Higgs dataset, we tested several values of k in $[10, 100]$, and several window sizes N in $[10^3, 10^6]$. For brevity we report only the results for $k = 20$, since the behaviors observed for the other values exhibit a similar pattern. For OUR- SLIDING and OUR- OBLIVIOUS, we set $\epsilon = 1$ and $\beta = 0.2$. For a fair comparison, we searched the parameter space of CSS- SLIDING so to determine a value of its parameter ϵ (the equivalent of our parameter β) so to enforce that the algorithms use approximately the same working memory. As a result, for CSS- SLIDING we set $\epsilon = 0.01$ which, in all of our tests, makes its working memory comparable yet slightly larger than the one used by OUR- SLIDING and OUR- OBLIVIOUS, which gives a competitive advantage to CSS- SLIDING in the comparison with respect to the approximation quality. Similarly, for Covertype we report experiments with $k = 20$ and window lengths in $[10^3, 3{\cdot}10^5]$. Due to the higher dimensionality of the dataset, which calls for a finer control on accuracy, we set $\epsilon = 0.5$ and $\beta = 0.1$ for both OUR- SLIDING and OUR- OBLIVIOUS, which brings us to set $\epsilon = 0.01$ for CSS- SLIDING to attain a similar, yet higher, memory usage of the latter w.r.t to the former algorithms.

The results are reported in the plots of Figs. 1, 2, 3, and 4 for the dataset Higgs, and in the plots of Figs. 5, 6, 7, and 8 for Covertype. In the plots, the blue lines refer to CSS- SLIDING, the orange lines to OUR- SLIDING, the yellow lines to OUR- OBLIVIOUS, and the purple ones to the execution of the sequential algorithm GON on the entire window. Each point in a plot represents an average over 10,000 consecutive windows.

The comparison of the algorithms' memory usage is reported in Figs. 1 and 5. As expected, the working memory required by both CSS- SLIDING and our algorithms is mostly insensitive to the the window length, while the memory usage of GON clearly grow linearly with it. Note that OUR- OBLIVIOUS maintains consistently less points in memory than OUR- SLIDING, as it does not maintain the data structures relative to infeasible estimates, rebuilding them on the fly when

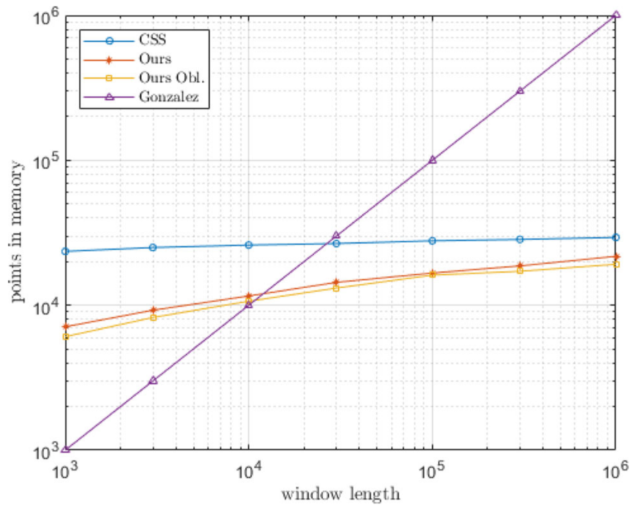

Fig. 1 Memory usage (Higgs)

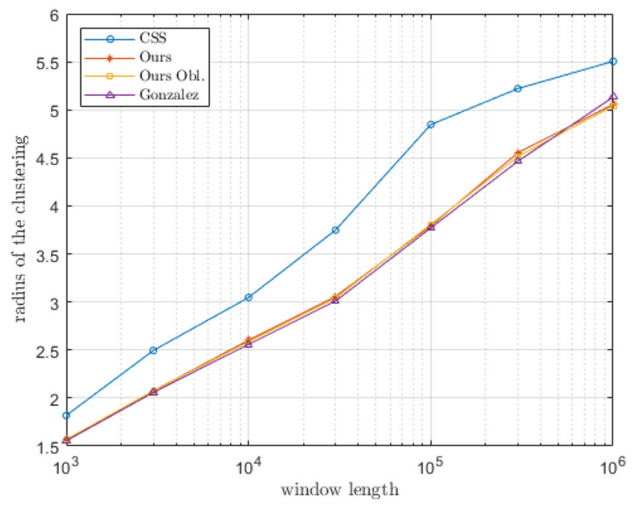

Fig. 2 Clustering radius (Higgs)

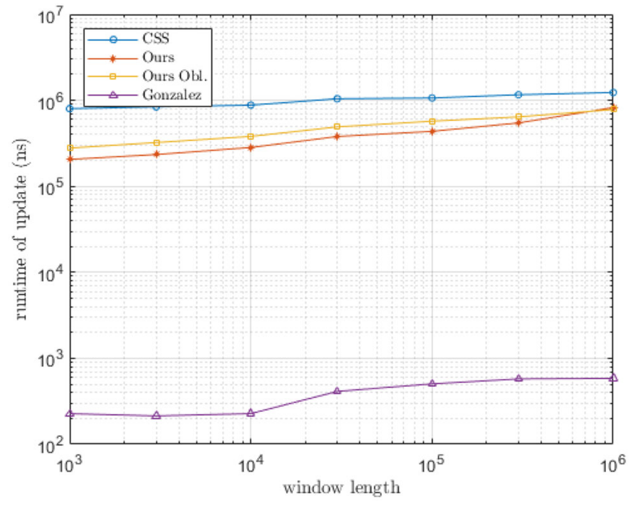

Fig. 3 Update time (Higgs)

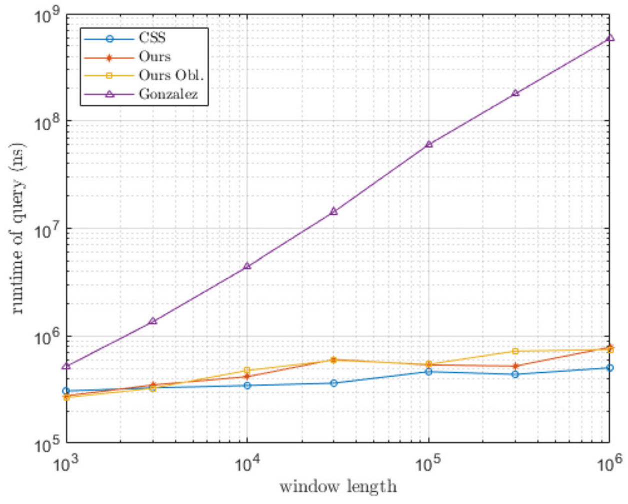

Fig. 4 Query time (`Higgs`)

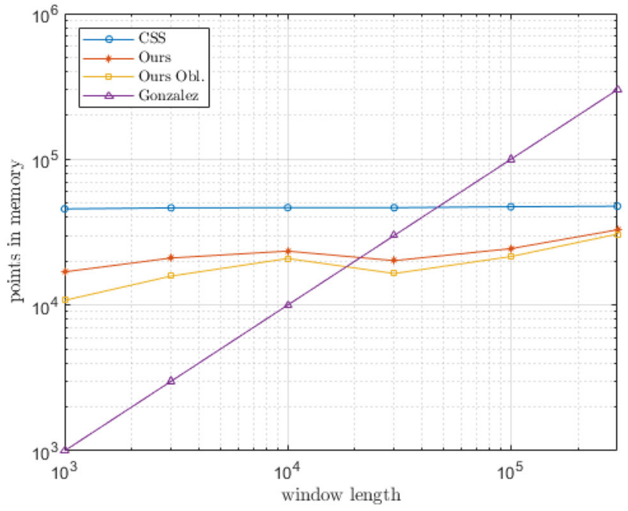

Fig. 5 Memory usage (`Covertype`)

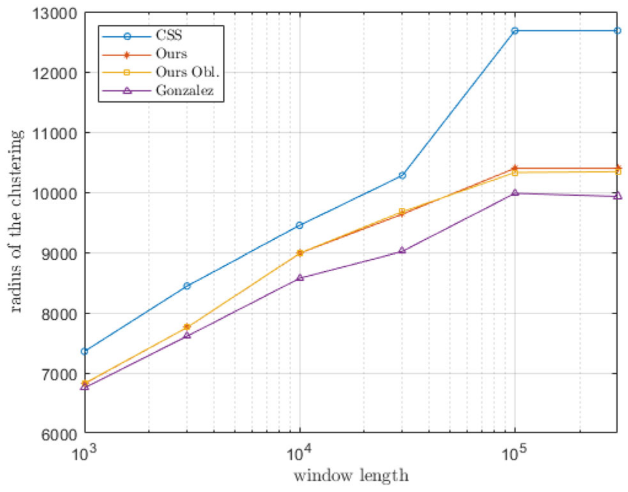

Fig. 6 Clustering radius (`Covertype`)

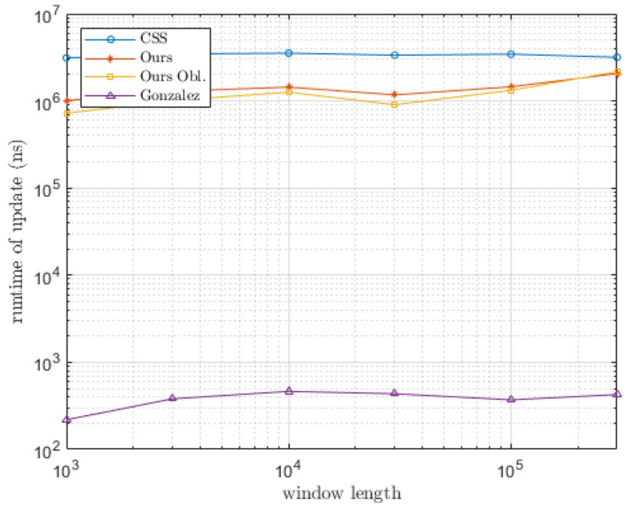

Fig. 7 Update time (`Covertype`)

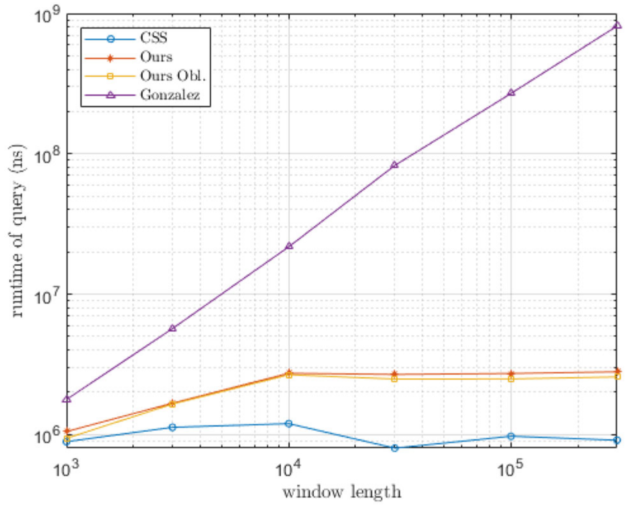

Fig. 8 Query time (`Covertype`)

needed. Figures 2 and 6 compare the clustering radii obtained by the four algorithms. Remarkably, on the `Higgs` dataset OUR- SLIDING and OUR- OBLIVIOUS, even for the relatively large value of $\epsilon = 1$, return a clustering whose radius essentially coincides with the one returned by running GON on the full window, and it is consistently and decidedly smaller than the one returned by CSS- SLIDING. Similarly on the `Covertype` dataset, the radii returned by our algorithms are consistently smaller than the ones returned by CSS- SLIDING, albeit not quite matching the radius of GON. By lowering the ϵ parameter one would obtain better results at the cost of an increased memory usage. The update time (Figs. 3 and 7), seems rather insensitive to N for both CSS- SLIDING OUR- SLIDING and OUR- OBLIVIOUS, while it is clearly negligible for GON, where it simply entails discarding the oldest point of the window and inserting the new one. Finally, as shown in

Figs. 4 and 8, the query times of CSS- SLIDING OUR- SLIDING and OUR- OBLIVIOUS are clearly much smaller than the one of GON, which grows linearly with the window length. The query times of OUR- SLIDING and OUR- OBLIVIOUS are comparable to those of CSS- SLIDING but slightly higher especially in the case of the `Covertype` dataset, which is conceivably due to its higher dimensionality.

Overall, the experiments provide evidence that, with respect to the state-of-the-art algorithm in [14], our algorithms OUR- SLIDING and OUR- OBLIVIOUS offer an approximate solution whose quality is much closer to the one guaranteed by best sequential algorithm run on the entire window, within comparable space and time budgets. Moreover the experiments show that the α-oblivious version of our algorithm consistently maintains less points in the working memory, without compromising neither execution times nor the quality of the solution.

5.3 Adaptiveness to dimensionality and aspect ratio

One of the prominent features of our algorithms is their ability to adapt to the inherent complexity of the window W_t, as captured by its dimensionality D_W, and, in the case of OUR- OBLIVIOUS, also by its aspect ratio α_W. The experiments reported below are aimed at testing such adaptiveness.

First, we investigate the dependency of the memory usage as a function of D_W. To this purpose, we used several artificial datasets consisting of points of $\mathbb{R}^{100}$ randomly extracted from the unit subcube $[0, 1]^d$ with d ranging from 1 up to 25 (by fixing $100 - d$ components to zero), making sure to fix the aspect ratio for all the datasets by periodically injecting "extreme" points in the stream[4]. For each of the datasets, we report the number of points maintained in the working memory by OUR- SLIDING (we omit for brevity the results of OUR- OBLIVIOUS, as they exhibit the same behavior) when used on a window of length $N = 10^5$ points with parameters $\beta = 0.1$, $\epsilon = 1$ and $k = 20$. All quantities are averaged over 10,000 consecutive windows and reported together with their 95% confidence intervals. As plotted in Fig. 9, for dimensions in the range [1, 15] the memory growth is clearly exponential with the dimension of the subspace, as suggested by the theory. For larger values of the subspace dimension, the growth slows down as the coreset size approaches the window length.

In a second experiment, we assessed the capability of our algorithms to adapt to the dimensionality of the data dynamically, as the window slides over sets of points of varying dimensionality. We generated an artificial dataset (DD) of random points in $\mathbb{R}^{100}$, where the first portion of points belongs to a subspace of dimension 1 (i.e., a line), the middle portion of points belongs to a subspace of dimension 10, and

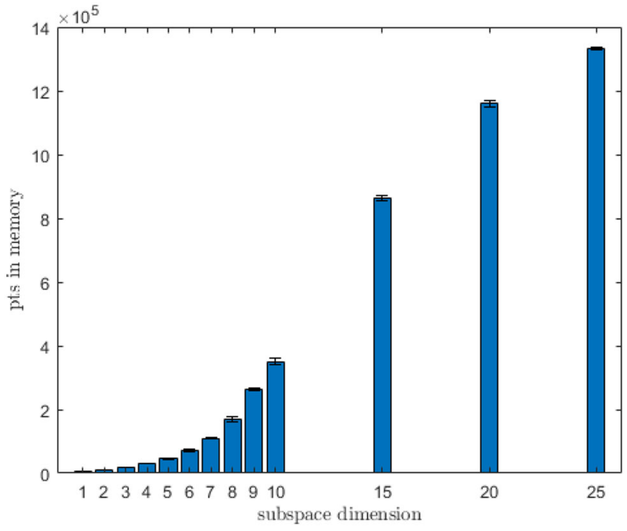

Fig. 9 Memory usage versus subspace dimension

the last portion belongs again to a line. In order to fix other features of the dataset that may influence memory usage, the points are generated so to make sure that the aspect ratio of all the windows is roughly the same, so that OUR- OBLIVIOUS is not influenced by the variability of α_W. As before, we set $\beta = 0.1$, $\epsilon = 1$ and $k = 20$ for both OUR- SLIDING and OUR- OBLIVIOUS. The windows length is $N = 10^4$. The memory usage of OUR- SLIDING and OUR- OBLIVIOUS, plotted in respectively Figs. 10 and 11, respectively, is low in the windows that cover the points in the first and third portion, and much higher in the windows that cover the points in the second portion, with a transition phase for the windows spanning the two types of subspaces. This property is very appealing, as the algorithms automatically adapt to the dimension D_W of the active window W_t to maintain as few points as possible, even without any prior knowledge on the doubling dimension. We observe that the higher level of noise in the plot for OUR- OBLIVIOUS is due to the higher variability of the range of guesses Γ_t (which mostly depends on the variability of r_t), as opposed to the fixed range Γ used by OUR- SLIDING.

Finally, we present some experiments on an artificial dataset (`alpha`) of random points in $\mathbb{R}^{100}$, where the aspect ratio varies substantially from window to window while the doubling dimension remains stable. In order to have multiple aspect ratios in the same dataset, we generated the points as follows. All the points have the first five coordinates distributed uniformly in some range and the others set to 0. In the first portion of the stream 90% of the points have coordinates in [0, 10] and 10% of the points have coordinates in [0, 100]. In the second portion of the stream 90% of the points have coordinates in [0, 0.01], hence the minimum distance among any two points will be around 10^3 times smaller than before, and 10% of the points have coordinates in [0, 10], hence the

[4] We remark that the doubling dimension of a subspace of $[0, 1]^d$ is linearly related to d.

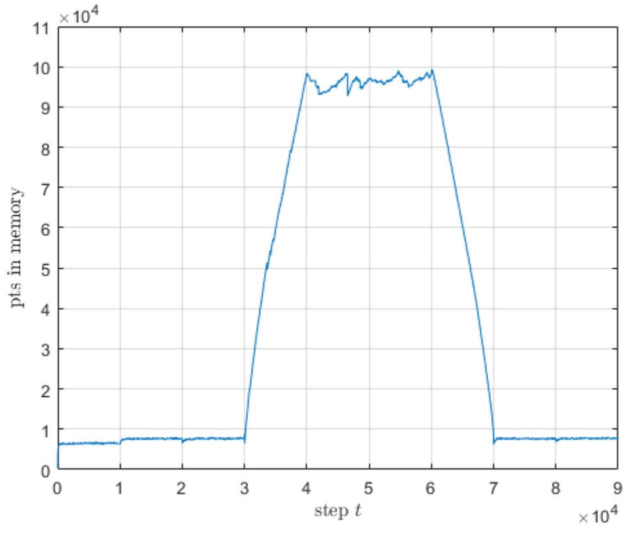

Fig. 10 OUR- SLIDING memory usage (DD)

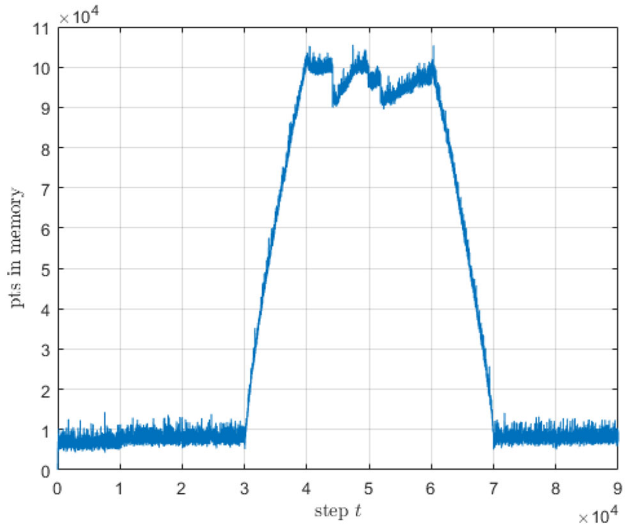

Fig. 11 OUR- OBLIVIOUS memory usage (DD)

diameter will be around 10 times smaller than before, for an overall factor 100 increase in the aspect ratio. Finally, the last part of the stream has the same distribution as the first one.

Once again we set $\beta = 0.1$, $\epsilon = 1$ and $k = 20$ for both OUR- SLIDING and OUR- OBLIVIOUS and set the window length $N = 10^4$. Figure 12a and b show the memory usage of the two algorithms, as a function of the step t, while Fig. 12c plots how the extremal values $r_t/2$ and $2M_t/\delta$ of range Γ_t (used by OUR- OBLIVIOUS to attain obliviousness to the aspect ratio) evolve with t. We first observe that the memory usage of OUR- OBLIVIOUS is slightly lower than the one of OUR- SLIDING. This feature, as seen in previous experiments, does not arise only in tailor-made artificial datasets but also in real-world ones (see Figs. 1 and 5), giving a competitive advantage to the oblivious version of the algorithm over the non-oblivious one. Somewhat surprisingly, however, the memory usage of OUR- SLIDING also seems to adapt to the shape of Γ_t although it uses a fixed range Γ for the guesses. This phenomenon is due to the fact that at each time t, for each guess $\gamma \in \Gamma - \Gamma_t$, the number of validation and coreset points maintained by OUR- SLIDING is rather small.

5.4 Comparison with the state-of-the-art diameter algorithm

As for the k-center algorithm, we implemented our diameter approximation algorithm, hereinafter referred to as OUR- DIAMETER, and we tested it against the sliding window algorithm in [14], which we will refer to as CSS- DIAMETER. For efficiency, we substituted the expensive quadratic-time naive computation of the exact diameter, wherever it was required, with the following greedy heuristic (referred to GREEDY- DIAMETER) proposed in [32]: starting from a random point p_0, select the point p_1 farthest away from p_0, then select the point p_2 farthest from p_1, and return $\text{dist}(p_1, p_2)$. While, theoretically, $\text{dist}(p_1, p_2)$ is only guaranteed to be within a factor 2 from the actual diameter, it has been observed that, in most cases, the difference between the two quantities is, in fact, very small [32]. In light of this fact, in the experiments we used GREEDY- DIAMETER both as a baseline to compute the diameter Δ_W of the window W, using the entire window as an input, and as a means of computing the diameter Δ_T of the coreset T. Thus, in the experiments we do not differentiate between the two variants to obtain Δ_T (exactly or approximately) described in Sect. 4.

For what concerns the `Higgs` dataset, we experimented with window lengths varying in $[10^3, 10^6]$ and parameters $\epsilon = 0.5$ and $\beta = 0.1$, for OUR- DIAMETER, and $\epsilon = 0.001$, for CSS- DIAMETER, so to make sure, for fairness, that CSS- DIAMETER requires approximately the same amount of working memory as OUR- DIAMETER (but not less). All the quantities in the plots are averaged over 10,000 consecutive windows. Figure 13, reports the memory usage of both algorithms together the (linear) memory usage of the baseline. Figure 14 compares the accuracy exhibited by the algorithms. Specifically, for each value of the window length the figure reports the ratio of the distance returned by each algorithm to the distance computed by GREEDY- DIAMETER when run on the entire window. Surprisingly, the solution of CSS- DIAMETER already exhibits rather good accuracy albeit, in theory, the algorithm only guarantees a $(3 + \epsilon)$ approximation. Our algorithm OUR- DIAMETER turns out slightly more accurate than CSS- DIAMETER, offering a good trade-off between the quality of the solution and the memory usage. We remark that CSS- DIAMETER delivers results of comparable quality also for larger values of ϵ (that is, smaller memory usage), hence it seems that the algorithm is less effective in exploiting larger memory budgets.

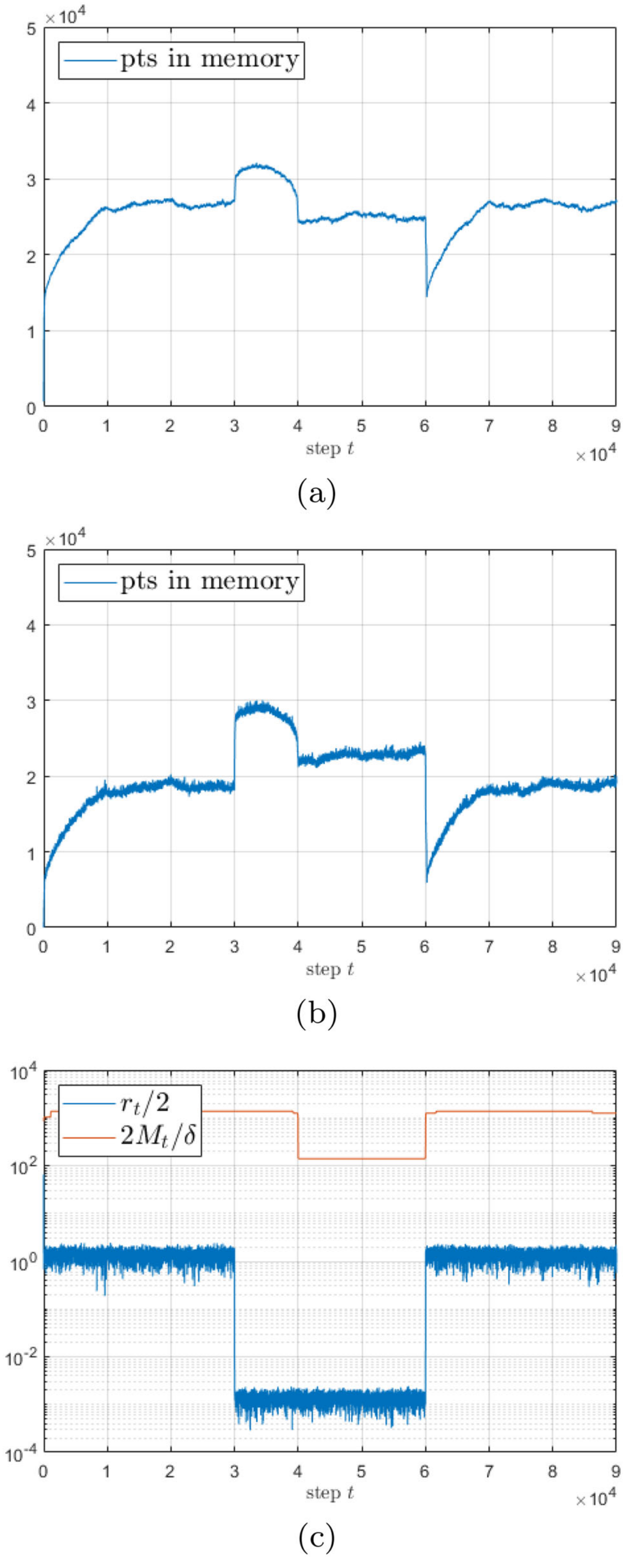

Fig. 12 Memory usage of OUR- SLIDING (**a**), OUR- OBLIVIOUS (**b**), and extremal values for Γ_t (**c**)

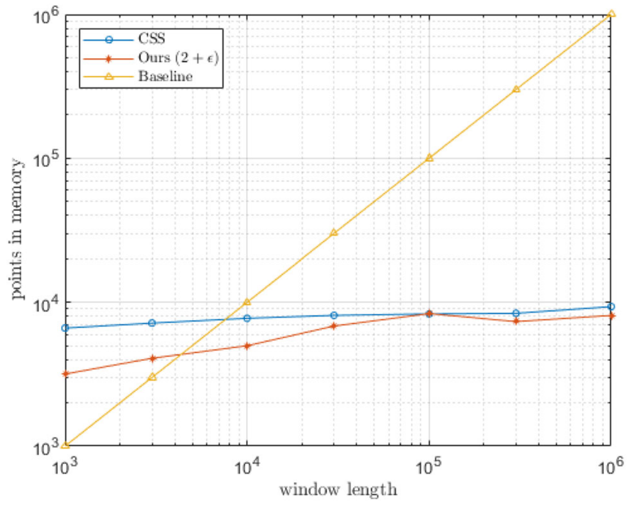

Fig. 13 Memory usage (Higgs)

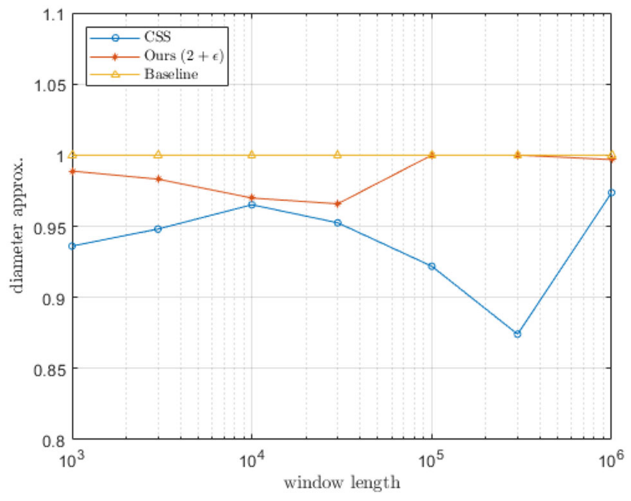

Fig. 14 Diameter approximation (Higgs)

For what concerns the Covertype dataset, we repeated the same experiment with window lengths varying in $[10^3, 2*10^5]$ and parameters $\epsilon = 1$ and $\beta = 1$, for OUR- DIAMETER, and $\epsilon = 0.001$, for CSS- DIAMETER, again to ensure, for fairness, that CSS- DIAMETER requires approximately the same amount of working memory. The results concerning the memory requirements and the approximation are reported in Figs. 15 and 16.

Due to higher dimensionality of the dataset we were forced to increase the values of ϵ and β for OUR- DIAMETER in order to keep the memory requirements reasonable and below the window length. Nonetheless the approximation quality featured by OUR- DIAMETER is superior to the one of CSS- DIAMETER and matches the one of the baseline for smaller window lengths, while it degrades for larger windows, remaining however always within 10% of the estimate provided by the baseline. This degradation is probably caused

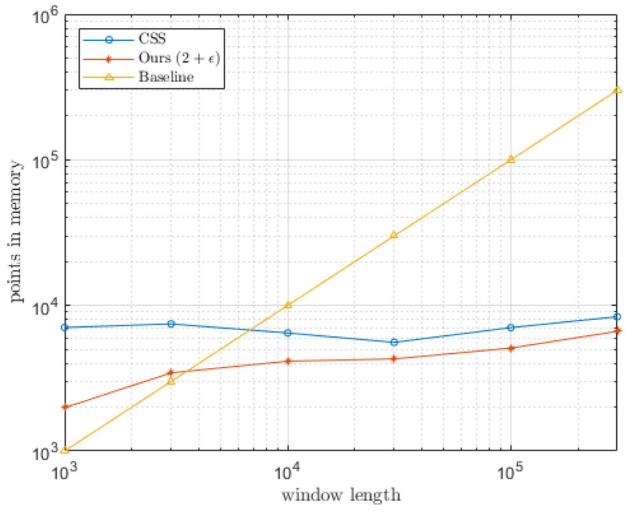

Fig. 15 Memory usage (`Covertype`)

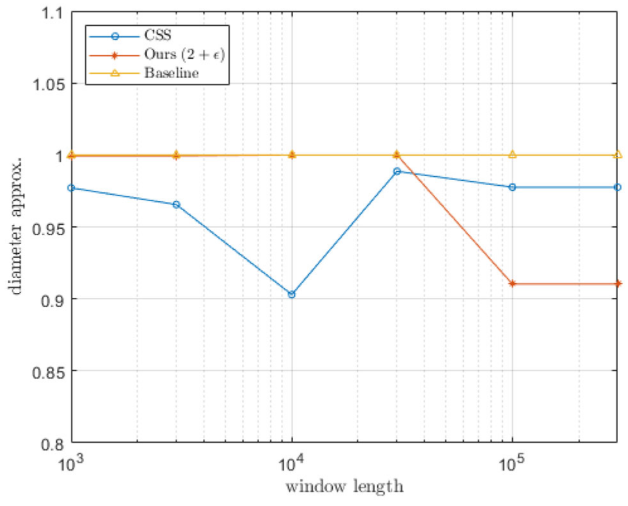

Fig. 16 Diameter approximation (`Covertype`)

by the high dimensionality of the dataset, which forced us to use larger values of ϵ and β to keep memory under control.

Regarding update and query times, for both the `Higgs` and `Covertype` datasets, we observed for all algorithms the same behaviors discussed for the k-center problem (plots are omitted for brevity). Namely, update and query times are independent of the window length and linear in the working memory, for both OUR- DIAMETER and CSS- DIAMETER. Also, the query times OUR- DIAMETER and CSS- DIAMETERare comparable, with the former being slightly larger for the higher dimensional `Covertype` dataset.

6 Conclusions

In this paper, we have shown how to attain coreset constructions yielding accurate streaming algorithms for the k-center and diameter estimation problems under the sliding window model. While the algorithms require very limited amounts of working memory for windows W of low doubling dimension D_W, the approach quickly degrades as D_W grows large. An interesting, yet challenging, research avenue is to investigate whether this steep dependence on D_W can be ameliorated by means of alternative techniques (e.g., the use of randomization).

Funding Open access funding provided by Università degli Studi di Padova within the CRUI-CARE Agreement. This work was supported, in part, by MIUR, the Italian Ministry of Education, University and Research, under PRIN Project n. 20174LF3T8 AHeAD (Efficient Algorithms for HArnessing Networked Data), and grant L. 232 (Dipartimenti di Eccellenza), and by the University of Padova under project SID 2020 RATED-X (Resource-Allocation TradEoffs for Dynamic and eXtreme data).

Code Availability The code for the experiments and is publicly available on GitHub (https://github.com/PaoloPellizzoni/AdaptiveCoreset).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ackermann, M.R., Blömer, J., Sohler, C.: Clustering for metric and nonmetric distance measures. ACM Trans. Algorithms **6**(4), 59:1-59:26 (2010)
2. Agarwal, P., Matoušek, J., Suri, S.: Farthest neighbors, maximum spanning trees and related problems in higher dimensions. Comput. Geom. **1**, 189–201 (1992)
3. Agarwal, P., Sharathkumar, R.: Streaming algorithms for extent problems in high dimensions. Algorithmica **72**, 83–98 (2015)
4. Awasthi, P., Balcan, M.: Center based clustering: a foundational perspective. In: Handbook of Cluster Analysis. CRC Press (2015)
5. Bateni, M., Esfandiari, H., Jayaram, R., Mirrokni, V.: Optimal fully dynamic k-centers clustering. Preprint ArXiv:2112.07050 (2021)
6. Borassi, M., Epasto, A., Lattanzi, S., Vassilvitskii, S., Zadimoghaddam, M.: Sliding window algorithms for k-clustering problems. In: Proceedings of NeurIPS, pp. 8716–8727 (2020)
7. Borassi, M., Epasto, A., Lattanzi, S., Vassilvitskii, S., Zadimoghaddam, M.: Better sliding window algorithms to maximize

subadditive and diversity objectives. In: Proceedings of ACM PODS, pp. 254–268 (2019)
8. Ceccarello, M., Pietracaprina, A., Pucci, G., Upfal E.: A practical parallel algorithm for diameter approximation of massive weighted graphs. In: Proceedings of IEEE IPDPS, pp. 12–21 (2016)
9. Ceccarello, M., Pietracaprina, A., Pucci, G.: Fast coreset-based diversity maximization under matroid constraints. In: Proceedings of ACM WSDM, pp. 81–89 (2018)
10. Ceccarello, M., Pietracaprina, A., Pucci, G.: Solving k-center clustering (with outliers) in MapReduce and streaming, almost as accurately as sequentially. PVLDB **12**(7), 766–778 (2019)
11. Ceccarello, M., Pietracaprina, A., Pucci, G., Upfal E.: Distributed graph diameter approximation. Algorithms. **13**(9), 216:1-216:23 (2022)
12. Chan, T., Sadjad, B.: Geometric optimization problems over sliding windows. Int. J. Comput. Geom. Appl. **16**(2–3), 145–158 (2006)
13. Chan, T.H.H., Guerqin, A., Sozio, M.: Fully dynamic k-center clustering. In: Proceedings of WWW, pp. 579–587 (2018)
14. Cohen-Addad, V., Schwiegelshohn, C., Sohler, C.: Diameter and k-center in sliding windows. In: Proceedings of ICALP, pp. 19:1–19:12 (2016)
15. Cole, R., Gottlieb, L.A.: Searching dynamic point sets in spaces with bounded doubling dimension. In: Proceedings ACM STOC, pp. 574–583 (2006)
16. Datar, M., Motwani, R.: The sliding-window computation model and results. In: Data Stream Management-Processing High-Speed Data Streams, pp. 149–165 (2016)
17. de Berg, M., Monemizadeh, M., Zhong, Y.: k-center clustering with outliers in the sliding-window model. In: Proceedings of ESA, pp. 13:1–13:13 (2021)
18. Feigenbaum, J., Kannan, S., Zhang, J.: Computing diameter in the streaming and sliding-window models. Algorithmica **41**(1), 25–41 (2005)
19. Epasto, A., Lattanzi, S., Vassilvitskii, S., Zadimoghaddam, M.: Submodular optimization over sliding windows. In: Proceedings of WWW, pp. 421–430 (2017)
20. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theoret. Comput. Sci. **38**, 293–306 (1985)
21. Goranci, G., Henzinger, M., Leniowski, D., Schulz, C., Svozil, A.: Fully dynamic k-center clustering in doubling metrics. arXiv:1908.03948 (2019)
22. Gottlieb, L.A., Kontorovich, A., Krauthgamer, R.: Efficient classification for metric data. IEEE Trans. Inform. Theory **60**(9), 5750–5759 (2014)
23. Guha, S.: Tight results for clustering and summarizing data streams. In: Proceedings of ICDT, pp. 268–275 (2009)
24. Gupta, A., Krauthgamer, R., Lee, J.R.: Bounded geometries, fractals, and low-distortion embeddings. In: Proceedings of IEEE FOCS, pp. 534–543 (2003)
25. Hennig, C., Meila, M., Murtagh, F., Rocci, R.: Handbook of Cluster Analysis. CRC Press, New York (2015)
26. Henzinger, M., Raghavan, P., Rajagopalan, S.: Computing on data streams. In: Proceedings of DIMACS Workshop on External Memory Algorithms, pp. 107–118 (1998)
27. Indyk, P.: Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In: Proceedings of ACM-SIAM SODA, pp. 539–545 (2003)
28. Kim, S.S.: Computing Euclidean k-center over sliding windows. arXiv:2001.01035 (2020)
29. Konjevod, G., Richa, A.W., Xia, D.: Dynamic routing and location services in metrics of low doubling dimension. In: Proceedings of DISC, pp. 379–393 (2008)
30. Korycki, L., Krawczyk, B.: Unsupervised drift detector ensembles for data stream mining. In: Proceedings of IEEE DSAA, pp. 317–325 (2019)
31. Leskovec, J., Rajaraman, A., Ullman, J.: Mining of Massive Datasets, 2nd edn. Cambridge University Press, Cambridge (2014)
32. Magnien, C., Latapy, M., Habib, M.: Fast computation of empirically tight bounds for the diameter of massive graphs. ACM J. Exp. Algorithmics **13**, 1.10:1–1.10:9 (2008)
33. McCutchen, R., Khuller, S.: Streaming algorithms for k-center clustering with outliers and with anonymity. In: Proceedings of APPROX-RANDOM, pp. 165–178 (2008)
34. Palmer, C., Gibbons, P., Faloutsos, C.: ANF: a fast and scalable tool for data mining in massive graphs. In: Proceedings of KDD. pp. 81–90 (2002)
35. Pelckmans, K.: Monitoring high-frequency data streams in FinTech: FADO Versus K-Means. IEEE Intell. Syst. **35**(2), 36–42 (2020)
36. Pellizzoni, P., Pietracaprina, A., Pucci, G.: Dimensionality-adaptive k-center in sliding windows. In: Proceedings of IEEE DSAA, pp. 197–206 (2020)
37. Pellizzoni, P., Pietracaprina, A., Pucci, G.: k-center clustering with outliers in sliding windows. Algorithms **15**, 7200 (2022)
38. Rehn, P., Ahmadi, Z., Kramer, S.: Forest of normalized trees: fast and accurate density estimation of streaming data. In: Proceedings of IEEE DSAA, pp. 199–208 (2018)
39. Shun, J.: An evaluation of parallel eccentricity estimation algorithms on undirected real-world graphs In: Proceedings of ACM KDD, pp. 1095–1104 (2015)
40. Snyder, L.: Introduction to facility location. In: Wiley Enciclopedia of Operations Research and Management Science. Wiley (2011)
41. Upadhyay, J.: Sublinear space private algorithms under the sliding window model. In: Proceedings of ICML, pp. 6363–6372 (2019)
42. Wang, Y., Li, Y., Tan, K.: Coresets for minimum enclosing balls over sliding windows. In: Proceedings of ACM KDD, pp. 314–323 (2019)