Dimensionality-adaptive k-center in sliding windows

Paolo Pellizzoni
Dept. of Information Engineering
University of Padova
Padova, Italy
paolo.pellizzoni@studenti.unipd.it

Andrea Pietracaprina
Dept. of Information Engineering
University of Padova
Padova, Italy
andrea.pietracaprina@unipd.it

Geppino Pucci
Dept. of Information Engineering
University of Padova
Padova, Italy
geppino.pucci@unipd.it

***Abstract*—In this paper we present a novel streaming algorithm for the k-center clustering problem for general metric spaces under the sliding window model. The algorithm maintains a small coreset which, at any time, allows to compute a solution to the k-center problem on the current window with an approximation quality that can be made arbitrarily close to the best approximation attainable by a sequential algorithm running on the entire window. Remarkably, the size of our coreset is independent of the window size and can be upper bounded by a function of k, of the desired accuracy, and of the doubling dimension of the metric space induced by the stream. For streams of bounded doubling dimension, the coreset size is merely linear in k. One of the major strengths of our algorithm is that it is fully oblivious to the doubling dimension of the stream, and it adapts to the characteristics of each individual window. Also, unlike previous works, the algorithm can be made oblivious to the aspect ratio of the metric space, a parameter related to the spread of distances. We also provide experimental evidence of the practical viability of the approach and its superiority over the current state of the art.**

***Keywords*-k-center clustering, data streams, sliding window model, coreset, doubling-dimension, approximation algorithms**

I. Introduction

Clustering is a fundamental primitive for data analysis, with applications in a variety of domains such as database search, bioinformatics, pattern recognition, networking, operation research, and many more [1], [2]. In this paper, we focus on the popular k-center variant which, given a set P of points from a metric space and a parameter $k < |P|$, requires to identify a set $S \subset P$ of k centers minimizing the maximum distance of any point from its closest center.

In several current application domains (e.g., social networks, online finance, online transaction systems), data are generated at a high rate in a continuous fashion, and their processing requires on-the-fly computation while maintaining in memory only a small portion of the data. 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 [3], [4]. In some prominent applications, it is important that older data in the stream (i.e., those outside a sliding window containing the N most recent data items) should be considered "stale" and should be disregarded in returning the desired solution. 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 with respect to the earlier ones. For this latter setting, an important variant of the streaming model, known as the *sliding window model*, was introduced in [5].

In this paper, we design, analyze, and experiment with a novel streaming algorithm for the k-center problem under the sliding window model. Our algorithm improves over the state of the art in several directions, as described in Subsection I-B.

A. Related Work

In the standard static sequential setting, it is well known that k-center is NP-hard, that it admits a 2-approximation algorithm, and that for any $\varepsilon > 0$ it is not $(2-\varepsilon)$-approximable unless P=NP [6].

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 [7] the authors developed a $(2+\varepsilon)$-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 entire cluster of x in time proportional to the cluster size. These results have been recently improved in [8] for spaces of constant doubling dimension. 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 [9], and, independently, Guha [10], presented algorithms which maintain a $(2+\varepsilon)$-approximation to the k-center problem for the entire set of points entered from the beginning of the stream, using working memory polynomial in k and $1/\varepsilon$. In the more restrictive sliding window setting, which is the focus of this paper, Cohen-Addad et al. [11] presented an algorithm which is able to compute a $(6+\varepsilon)$-approximation to the k-center problem for the current window, from only $O\left(k\varepsilon^{-1}\log\alpha\right)$ points stored in the working memory, where α is the *aspect ratio* of the stream, that is, the ratio between the maximum and minimum distance between any two points of the stream. At any time, the algorithm requires $O\left(k\varepsilon^{-1}\log\alpha\right)$ update time for handling the new point arrived from the stream, and $O\left(k^2\varepsilon^{-1}\log\alpha\right)$ 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. 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 of less than 4 requires working memory of size $\Omega\left(N^{1/3}\right)$, where N is the window size. In a recent unpublished manuscript, Kim [12] improved the result in [11] for Euclidean spaces, by presenting an algorithm which attains a $(2+2\sqrt{3}+\varepsilon)$-approximation through a coreset-based approach. The algorithm makes crucial use of the space dimensionality and is, thus, not immediately portable to non-Euclidean spaces. The author also claims that a $(2+\varepsilon)$-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.

B. Our contribution

In this work we present a streaming algorithm for the k-center problem under the sliding window model. At the heart of our algorithm is a coreset construction that takes inspiration from the one used by the algorithm in [11]. More specifically, our algorithm employs the data structures used in [11] to obtain a reasonable estimate of the optimal clustering radius. These structures are paired with additional ones which leverage the estimate of the radius to maintain a coreset containing better representatives for the points of the current window. The working memory used by our algorithm is analyzed as a function of a precision parameter ε, related to the desired approximation guarantee, and of the *doubling dimension* D of the metric space induced by the stream. The doubling dimension, which is formally defined in Section II, generalizes the notion of Euclidean dimensionality, and, as our results show, is related to the increasing difficulty of spotting good clusterings when its value grows.

For any fixed $\varepsilon > 0$, at any time t our algorithm is able to return a $(2+\varepsilon)$-approximate solution to the k-center problem for the current window W, using working memory $O\left(k\log(\alpha)(c/\varepsilon)^D\right)$, where α is the aspect ratio of the stream (i.e., the ratio between the maximum and the minimum distance of points of the stream), $c > 1$ is a suitable constant, and D is the doubling dimension of the stream. The update time required to handle each point is linear in the working memory size, while the query time to return the solution for the current window is subquadratic in the working memory size. (See Theorems 2 and 3 for precise bounds on the working memory and on the running times.) Observe that both working memory and time requirements are independent of the window size, and that, for constant ε and D, they grow asymptotically only as a function of k and α. Moreover, the approximation ratio of our algorithm can be made arbitrarily close to 2, which is the best approximation attainable by any polynomial-time sequential algorithm run on the entire window with unbounded memory.

The main improvements of our algorithm with respect to the state-of-the-art for general metric spaces [11] are:

- The approximation ratio drops from $6+\varepsilon$ to $2+\varepsilon$, with a moderate increase in the working memory and time requirements for low-dimensional streams. Moreover, our result shows that the lower bound on the working memory size proved in [11] can be beaten when the doubling dimension of the stream is small.
- Our algorithm is *fully oblivious* to the aspect ratio α and to the doubling dimension D, in the sense that these values are not used explicitly by the algorithm but they are only employed to analyze their space and time performance. This is a very desirable feature since, in practice, estimates for α and D are difficult to obtain. We remark that the algorithm in [11] exhibits the same performance for all values of D and, also, it requires prior knowledge of α.

Finally, as a proof of concept, we implemented our algorithm and the one by [11], and compared their performance. The experiments provide clear evidence that, when endowed with the same amount of working memory, our algorithm yields decidedly better approximation with comparable update and query times. Moreover, the solution quality provided by our algorithm is almost indistinguishable from the one of the 2-approximate sequential algorithm run on the entire window.

C. Organization of the paper

The rest of the paper is organized as follows. Section II defines the problem formally, and introduces a number of technical notions which will be used throughout the paper. Sections III and IV contain, respectively, the description and the analysis of our algorithm, under the assumption that the aspect ratio α is known, while Section V shows how to make the algorithm oblivious to α. Section VI presents the experimental results. Finally, Section VII offers some concluding remarks.

II. 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. 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 algorithm by Gonzalez [6] (dubbed GON in the rest of the paper), provides a sequential 2-approximation to the k-center problem and runs in $O(Nk)$ time, where $N = |W|$ is the input size. The following useful fact is proved in [13]:

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

Fact 1 states that GON, when run on a 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.

In the standard *streaming* framework [3], [4] the computation is performed by a single processor with a small working memory, and the input is provided as a continuous stream of items (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 size*, is a predetermined value. 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 size.

In this paper, we present a streaming algorithm for the k-center problem under the sliding window setting. Our algorithm uses a coreset-based strategy and, at any time t, it is able to extract a coreset T from the information stored in its working memory, such that a $(2+\varepsilon)$-approximate solution to the k-center problem for the pointset W_t can be efficiently obtained by running GON on T, where ε is a user-defined accuracy parameter. The coreset T used in our approach obeys to the following rigorous definition:

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

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 $OPT_{k,W}$.

The time and space performance of our algorithm will be analyzed in terms of the dimensionality of the points in the input stream. Since we target the applicability of our algorithm to arbitrary metric spaces, we will make use of the following, general notion of dimensionality. Let S denote a (possibly unbounded) set of points from a metric space. For any $x \in S$ and $r > 0$, let the *ball of radius r centered at x*, denoted as $B(x, r)$, be the subset of points of S at distance at most r from x. The *doubling dimension* of S is the smallest value D such that any ball $B(x, r)$, with $x \in S$, is contained in the union of at most 2^D balls of radius $r/2$ suitably centered at points of S. The following important fact, which we will use in the analysis, was proved in [14]:

Fact 2. *Let S be a set of points from a metric space and let $Y \subseteq S$ be such that any two distinct points $a, b \in Y$ have pairwise distance $\text{dist}(a, b) \geq r$. If S has doubling dimension D, then for every $R \geq r$ and any point $x \in S$, 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 input stream*, 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 [15], clustering [13], [16], nearest neighbor search [17], and machine learning [18].

III. Algorithm presentation

Consider a continuous stream S of points from a metric space with distance function $\text{dist}(\cdot,\cdot)$, a window size N, and a target number k of cluster centers. To simplify the presentation of the algorithm, we first make the assumption that the values *minDist* and *maxDist*, denoting, respectively, the minimum and maximum distances between any two distinct points of S, hence the aspect ratio $\alpha = maxDist/minDist$, are known to the algorithm. In Section V, we will show how this assumption can be removed, which is one of the

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

improvements of our algorithm with respect to the one in [11].

For each point p we define its *time-to-live* (TTL), denoted by 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 size, and, from that time on, TTL(p) decreases of one unit at every new arrival. To avoid continuous 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 TTL$(p)=0$. In 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

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

and note that $|\Gamma| = O\left(\log_{1+\beta}\alpha\right)$. As in [11], 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 OPT$_{k,W}$, and *coreset points* (*c-points* for short) which are used to actually extract the solution.

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 maintained according to a similar organization within sets A_γ, R_γ and O_γ. The sets of validation points are analogous to those used in [11]. 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 OPT$_{k,W}$ (that is, OPT$_{k,W}\le\gamma$). We say that a point p is *v-attracted* by $v\in AV_\gamma$ if dist$(p,v)<2\gamma$. The set RV_γ (*representative v-points*) contains a representative repV$_\gamma(v)$ for each $v\in AV_\gamma$, which is the newest point (that is, the point with the largest TTL) among those v-attracted by v. When v expires, its representative repV$_\gamma(v)$ becomes an *orphan*, and it is moved to the set OV_γ (*orphan v-points*).

Let $\varepsilon>0$ 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 can provide an ε-coreset for the current window. Let $\delta=\varepsilon/(1+\beta)$. 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 dist$(p,a)<\delta\gamma/2$. 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 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 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 $q=$ repV$_\gamma(v)$ (resp., $q=$ repC$_\gamma(a)$).

At every time step, all copies of a number of points, including all copies of the one 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 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 and uses, as a subroutine, INSERTVALIDATION(p,γ); finally, QUERY$()$, if invoked at time t, returns the coreset where algorithm GON can be run.

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

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

```
QUERY()
1  for increasing values of $\gamma \in \Gamma$ such that $|AV_\gamma| \le k$
2      $C = \emptyset$
3      for $p \in AV_\gamma \cup OV_\gamma \cup RV_\gamma$
4          if $\text{dist}(p, C) > 2\gamma$
5              $C = C \cup \{p\}$
6      if $|C| \le k$
7          $\hat{\gamma} = \gamma$
8          break;
9  // now $\hat{\gamma}$ is the smallest radius such that the clustering
   requires at most k centers
10 return $R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$
```

IV. Algorithm analysis

Consider an input stream S and suppose that Procedure UPDATE(p) is applied to any point $p \in S$ when it arrives. In this section, we show that, at any time, invoking Procedure QUERY (after UPDATE(p) has finished processing the last point p) returns an ε-coreset for the current window W and that by running the 2-approximation algorithm GON for k-center on the coreset, a $(2 + \varepsilon)$-approximate solution for W is obtained. Moreover, we will analyze the amount of working memory the time required to process each point of the stream. Space and time bounds will be expressed in terms of the various parameters involved and of the doubling dimension of S.

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

Lemma 1. *Let W denote the set of the last N points arrived in the stream. For every $\gamma \in \Gamma$, the following invariants hold at the end of each execution of Procedure* UPDATE(p).

1) *If $|AV_\gamma| \le k$, the following two inequalities hold:*
   a) $\max_{q \in W} \text{dist}(q, R_\gamma \cup O_\gamma) \le \delta\gamma$;
   b) $\max_{q \in W} \text{dist}(q, RV_\gamma \cup OV_\gamma) \le 4\gamma$.
2) *If $|AV_\gamma| > k$, the following two properties hold:*
   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 [11], 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 that 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. $\le \delta\gamma/2$) from q when q arrived. 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 12. If $|AV_\gamma| \le k$ before UPDATE(p) starts, it stays this way after Line 12, since Lines 1-12 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 10 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 it 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 $|AV_\gamma| > k$ 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 12, 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 12, 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 17. First, consider the case $EV = \emptyset$. Then INSERTVALIDATION is invoked to insert p in AV_γ. If at the end of INSERTVALIDATION we have $|AV_\gamma| \leq k$, then Invariant 1 holds since the call does not delete any point. Otherwise, at the end of INSERTVALIDATION, $|AV_\gamma| = k+1$ 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 16 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 $\text{repV}(v)$, with $v \in EV$, with the new point p. Note that both $\text{repV}(v)$ and p are *v-attracted* by the same point v, so, all points with the same v-attractor as $\text{repV}(v)$ 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 24. If $E = \emptyset$, then no point is deleted in Lines 18-24, thus the two invariants will hold. Otherwise, if $E \neq \emptyset$, we replace each representative c-point $\text{repC}(a), a \in E$ with the new point p. Since $\text{repC}(a)$ and p are *c-attracted* by the same point a, all points with the same c-attractor as $\text{repC}(a)$ 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 $\text{dist}(p, RV_\gamma) = 0$. Else, there exists a point $v \in AV_\gamma$ such that $\text{dist}(p, RV\gamma) \leq \text{dist}(p, v) + \text{dist}(v, RV_\gamma) \leq 4\gamma$. Similarly, it must hold that either $\text{dist}(p, R\gamma) = 0$ (case $E = \emptyset$) or $\text{dist}(p, R_\gamma) \leq \delta\gamma$ (case $E \neq \emptyset$), and the two invariants follow.

■

The next lemma shows that Procedure QUERY returns an ε-coreset of the current window.

Lemma 2. *Let* CS_W *be the set of points returned by Procedure* QUERY*, and let* W *be the current window. Then* CS_W *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 $\gamma < \text{OPT}_{k,W}$. Moreover, since Γ contains guess $\gamma \geq maxDist \geq \text{OPT}_{k,W}$, the procedure will always determine a minimum guess $\hat{\gamma}$ such that both $|AV_{\hat{\gamma}}| \leq k$ and $|C| \leq k$. Then, since Γ samples the interval $[minDist, maxDist]$ with a geometric progression of common ratio $(1+\beta)$, we obtain that $\hat{\gamma}/(1+\beta) < \text{OPT}_{k,W}$. Also, since $|AV_{\hat{\gamma}}| \leq k$, Invariant 1(a) ensures that

$$\max_{p \in W} \text{dist}(p, \text{CS}_W) < \delta\hat{\gamma} = \varepsilon\hat{\gamma}/(1+\beta) < \varepsilon \cdot \text{OPT}_{k,W},$$

and the lemma follows. ■

The next theorem establishes the approximation factor of our algorithm.

Theorem 1. *Let* CS_t *be the set of points returned by Procedure* QUERY *if invoked at time t, and let* W *be the current window. Then, by running Algorithm* GON *on* CS_t *we obtain a* $(2+\varepsilon)$*-approximate solution for the k-center problem on* W*.*

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

■

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*

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

points, overall.

Proof: Consider an arbitrary time t. 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 [11], 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 $\text{TTL}(v_{i+k+1}) > \text{TTL}(\text{repV}_\gamma(v_i)) \geq \text{TTL}(v_i)$. Consider two cases. If v_i expires before v_{i+k+1} enters the window, then $\text{TTL}(v_{i+k+1}) > \text{TTL}(\text{repV}_\gamma(v_i))$ because $\text{repV}_\gamma(v_i)$ 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 $\text{TTL}(v_{i+k+1}) > \text{TTL}(\text{repV}_\gamma(v_i))$ because $\text{repV}_\gamma(v_i)$ 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 11 of INSERTVALIDATION. This shows that $|OV_\gamma| \leq k+1$.

Next, we show that $|A_\gamma \cup R_\gamma \cup O_\gamma| \leq 6(k+1)(32/\delta)^D$, where D is the doubling dimension of S. 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 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 $\mathrm{TTL}(q) \geq \min_{v\in AV_\gamma} \mathrm{TTL}(v)$ 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 $(32/\delta)^D$ points of A_γ, so $|A_\gamma| \leq 2(k+1)(32/\delta)^D$. 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' 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 $\mathrm{TTL}(a_{i+k'+1}) > \mathrm{TTL}(\mathrm{repC}_\gamma(a_i)) \geq \mathrm{TTL}(a_i)$. 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, $\mathrm{TTL}(a_{i+k'+1}) > \mathrm{TTL}(\mathrm{repC}_\gamma(a_i))$ because $\mathrm{repC}_\gamma(a_i)$ must have entered the window before a_i expired or got deleted, which means that $\mathrm{repC}_\gamma(a_i)$ 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 $\mathrm{TTL}(v_j)$, thus it cannot be in memory at time t because it either expired or has been deleted by Line 11 of INSERTVALIDATION. This shows that there can be at most k' points in O_γ. ■

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

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

while Procedure QUERY$()$ *runs in time*

$$O\left(k^2 \cdot \log\left(\frac{\log(\alpha)}{\log(1+\beta)}\right) + k \cdot \left(\frac{32(1+\beta)}{\varepsilon}\right)^D\right).$$

Proof: The time complexity of UPDATE(p) is dominated by the construction of the sets EV and E for each γ (Lines 11 and 12), 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 $\hat{\gamma}$ can be easily accomplished in $O\left(k^2 \log(\alpha)/\log(1+\beta)\right)$ time. In fact, by using binary search over the values $\gamma \in \Gamma$, the time can be reduced to $O\left(k^2 \log(\log(\alpha)/\log(1+\beta))\right)$. Finally, once $\hat{\gamma}$ has been found, returning $R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ takes time proportional to their size, which is $O\left(k(32(1+\beta)/\varepsilon)^D\right)$ as argued in the proof of Theorem 2. ■

The following corollary, which summarizes the main features of our algorithm, is an immediate consequence of Theorems 1, 2 and 3.

Corollary 1. *Consider a stream S of points from a metric space and a sliding window of length N. Let D be the doubling dimension of S and let α be the ratio between the maximum and minimum distance of two points of S. For fixed parameters $\varepsilon, \beta > 0$, at any time our algorithm requires working memory of size $M = O\left(k \cdot (\log(\alpha)/\log(1+\beta))(32(1+\beta)/\varepsilon)^D\right)$, processes each point $p \in S$ in time $O(M)$, and at any time t is able to return a $(2+\varepsilon)$-approximation to the k-center problem for the current window in time $O\left(k^2 \cdot \log(\log(\alpha)/\log(1+\beta)) + k \cdot (32(1+\beta)/\varepsilon)^D\right)$.*

V. Obliviousness to α

For convenience, the algorithm presented in Section III assumed the knowledge of α, that is, the ratio between the maximum and minimum distance of any two distinct points in the stream. In this section, we show how this assumption can be removed with some slight modifications to the algorithm, which are described below.

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 $(p_{t-k}, \ldots, p_{t-1}, p_t)$, and observe that $r_t \leq \mathrm{OPT}_{k,W}$ for the current window W. 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 the algorithm to maintain the value $M_t = 2\max_{1<i\leq t} \text{dist}(p_i, p_1)$, which is easly seen to upper bound the diameter (i.e., the maximum pairwise distance) of all points arrived so far, to within a factor at most 2.

Let the values β, ε and $\delta = \varepsilon/(1+\beta)$ be defined as in Section III, and assume that $\delta \leq 4$ (in fact, larger values of δ would be uninteresting in 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 following claim shows that at any step t, it is sufficient that the algorithm maintains structures for guesses γ belonging to Γ_t.

Claim 1. *Consider the non-oblivious algorithm presented in Section III. 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: Consider a value $\gamma < \min \Gamma_t$ 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 main for loop of QUERY. Let $\gamma_{\max} = \max \Gamma_t$. 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 up to point p_t. ■

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 up to some time $t-1 > k$ and consider the arrival of p_t. First, the algorithm computes the new values r_t and M_t 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 $OV_\gamma = O_\gamma = \emptyset$. Observe that, for these values of γ, these would have been the sets maintained by the non-oblivious algorithm at the end of step $t-1$ since all the last $k+1$ points of the stream up to p_{t-1} would have made their way into AV_γ being all at pairwise distance $> 2\gamma$. 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 $OV_\gamma = O_\gamma = \emptyset$. Observe that, 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 $\text{dist}(q, RV_\gamma) = \text{dist}(q, R_\gamma) = \text{dist}(q, p_{t-1}) \leq M_{t-1} \leq \delta\gamma/2 \leq 2\gamma \leq 4\gamma$, 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 $t-1$. Finally, the algorithm runs UPDATE(p_t) limiting the main loop to values $\gamma \in \Gamma_t$.

The above discussion suffices to prove that the α-oblivious algorithm described above yields the same result of Corollary 1 with the only difference that an additive term $O\left(k^2\right)$ is required in the processing time of each point of the stream.

VI. Experiments

As a proof of concept to assess the practical viability of our approach, we designed a set of experiments to compare approximation ratio, execution time, and memory usage of our algorithm against the state-of-the-art algorithm in the published literature [11]. We devised an implementation of our algorithm, which we will refer to as OUR-SLIDING, and the sliding window algorithm by Cohen-Addad et al., hereinafter referred to as CSS-SLIDING. For the sake of a fair comparison, since CSS-SLIDING is not oblivious to α, OUR-SLIDING implements the non-oblivious version of our approach. Due to the NP-hardness of the k-center problem, it is impractical to compute the optimal solution for each window so to measure the exact approximation factor of the solutions returned by two algorithms. As a workaround, we compared these solutions against the one obtained by running the sequential algorithm GON on the entire window.

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. The code for the experiments on the datasets is available at https://github.com/PaoloPellizzoni/CoresetSlidingWindows.

We experimented with the `Higgs` dataset[2], which 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; for the sake of this experiment we considered only these seven "high-level" features, and used Euclidean distance.

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, we set $\varepsilon = 1$ and $\beta = 0.1$. 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 two algorithms use the same working memory. As a result, we set $\varepsilon = 0.01$ which, in all of our tests, makes the working memory of CSS-SLIDING comparable yet slightly larger than the one used by OUR-SLIDING, which gives a competitive advantage to CSS-SLIDING in the

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

comparison with respect to the approximation quality. The results are reported in the plots of Fig. 1, 2, 3, and 4. In each plot, the blue line corresponds to CSS-SLIDING, the orange line corresponds to OUR-SLIDING, and the yellow line corresponds to the execution of the sequential algorithm GON on the entire window. All the quantities are averaged over 1000 consecutive windows.

The comparison of the algorithms' memory requirements is reported in Fig. 1. As expected, the working memory required by both CSS-SLIDING and OUR-SLIDING does not seem to be significantly affected by the window size while the memory requirements of GON grow linearly with it. Fig. 2 compares the clustering radii obtained by the three algorithms. Remarkably, OUR-SLIDING, even for the relatively large value $\varepsilon = 1$, returns 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. The update time (Fig. 3), seems rather insensitive to N for both CSS-SLIDING and OUR-SLIDING, 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 Fig. 4, the query times of CSS-SLIDING and OUR-SLIDING are comparable while, clearly, the one of GON is much higher and grows linearly with the window size.

Overall, the experiments provide evidence that, with respect to the state-of-the-art algorithm in [11], our algorithm OUR-SLIDING offers an approximate solution that matches almost perfectly the one returned by best sequential algorithm run on the entire window, within the same space and time budgets.

Finally, we wish to point out that we also implemented the α-oblivious version of the algorithm and ran the same experiments described in this section on additional datasets, obtaining similar results to the one presented. A full account of these additional experiments will be provided in the full version of this paper.

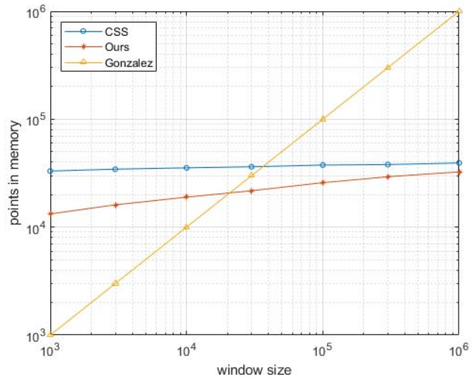

Figure 1: Comparison of memory requirements

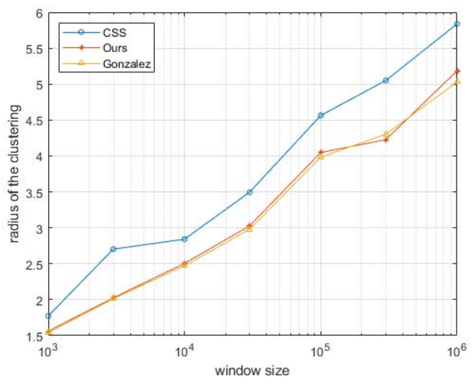

Figure 2: Comparison of clustering radii

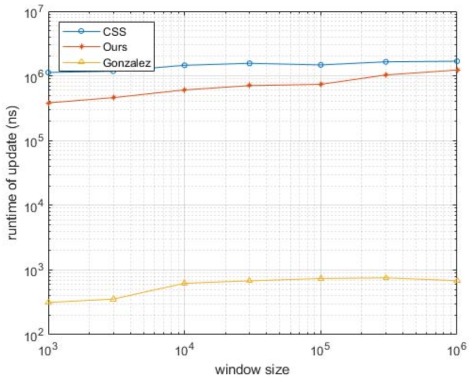

Figure 3: Comparison of update times

VII. Conclusions

In this paper, we have shown how to attain an improved coreset construction yielding an accurate streaming algorithm for the k-center problem under the sliding window model. While the algorithm exhibits very reasonable working memory bounds for streams of low doubling dimension D, the approach quickly degrades as D grows large. An interesting, yet challenging, research avenue is to investigate whether this steep dependence on D can be ameliorated by means of alternative techniques (e.g., the use of randomization).

Aknowledgements

This research was funded 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: Resource-Allocation Tradeoffs for Dynamic and Extreme Data".

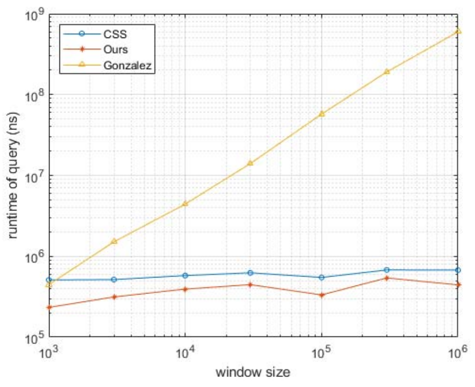

Figure 4: Comparison of query times

References

[1] C. Hennig, M. Meila, F. Murtagh, and R. Rocci, *Handbook of cluster analysis*. CRC Press, 2015.

[2] P. Awasthi and M. Balcan, "Center based clustering: A foundational perspective," in *Handbook of cluster analysis*. CRC Press, 2015.

[3] M. Henzinger, P. Raghavan, and S. Rajagopalan, "Computing on Data Streams," in *Proc. DIMACS Workshop on External Memory Algorithms*, 1998, pp. 107–118.

[4] J. Leskovec, A. Rajaraman, and J. Ullman, *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press, 2014.

[5] M. Datar and R. Motwani, "The sliding-window computation model and results," in *Data Stream Management - Processing High-Speed Data Streams*, 2016, pp. 149–165.

[6] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293 – 306, 1985.

[7] T.-H. H. Chan, A. Guerqin, and M. Sozio, "Fully Dynamic k -Center Clustering," in *Proc. TheWebConf*, 2018, pp. 579–587.

[8] G. Goranci, M. Henzinger, D. Leniowski, C. Schulz, and A. Svozil, "Fully dynamic k-center clustering in doubling metrics," *arXiv:1908.03948*, 2019.

[9] R. McCutchen and S. Khuller, *Streaming Algorithms for k-Center Clustering with Outliers and with Anonymity*, 2008, pp. 165–178.

[10] S. Guha, "Tight results for clustering and summarizing data streams," in *Proc. ICDT*, 2009, p. 268–275.

[11] V. Cohen-Addad, C. Schwiegelshohn, and C. Sohler, "Diameter and k-Center in Sliding Windows," in *Proc. ICALP*, 2016, pp. 19:1–19:12.

[12] S.-S. Kim, "Computing euclidean k-center over sliding windows," *arXiv:2001.01035*, 2020.

[13] M. Ceccarello, A. Pietracaprina, and G. Pucci, "Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially," *PVLDB*, vol. 12, no. 7, pp. 766–778, 2019.

[14] A. Gupta, R. Krauthgamer, and J. R. Lee, "Bounded geometries, fractals, and low-distortion embeddings," in *Proc. IEEE FOCS*, 2003, pp. 534–543.

[15] G. Konjevod, A. W. Richa, and D. Xia, "Dynamic routing and location services in metrics of low doubling dimension," in *Proc. DISC*, 2008, pp. 379–393.

[16] M. R. Ackermann, J. Blömer, and C. Sohler, "Clustering for metric and nonmetric distance measures," *ACM Trans. Algorithms*, vol. 6, no. 4, pp. 59:1–59:26, 2010.

[17] R. Cole and L.-A. Gottlieb, "Searching dynamic point sets in spaces with bounded doubling dimension," in *Proc. ACM STOC*, 2006, pp. 574–583.

[18] L.-A. Gottlieb, A. Kontorovich, and R. Krauthgamer, "Efficient classification for metric data," *IEEE Trans. Information Theory*, vol. 60, no. 9, pp. 5750–5759, 2014.