Article

k-Center Clustering with Outliers in Sliding Windows

Paolo Pellizzoni, Andrea Pietracaprina and Geppino Pucci *

Department of Information Engineering, University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy; paolo.pellizzoni@studenti.unipd.it (P.P.); andrea.pietracaprina@unipd.it (A.P.)
* Correspondence: geppino.pucci@unipd.it

Abstract: Metric *k*-center clustering is a fundamental unsupervised learning primitive. Although widely used, this primitive is heavily affected by noise in the data, so a more sensible variant seeks for the best solution that disregards a given number z of points of the dataset, which are called outliers. We provide efficient algorithms for this important variant in the streaming model under the sliding window setting, where, at each time step, the dataset to be clustered is the window W of the most recent data items. For general metric spaces, our algorithms achieve $O(1)$ approximation and, remarkably, require a working memory linear in $k + z$ and only logarithmic in $|W|$. For spaces of bounded doubling dimension, the approximation can be made arbitrarily close to 3. For these latter spaces, we show, as a by-product, how to estimate the effective diameter of the window W, which is a measure of the spread of the window points, disregarding a given fraction of noisy distances. We also provide experimental evidence of the practical viability of the improved clustering and diameter estimation algorithms.

Keywords: *k*-center with outliers; effective diameter; big data; data stream model; sliding windows; coreset; doubling dimension; approximation algorithms

check for updates

Citation: Pellizzoni, P.; Pietracaprina, A.; Pucci, G. *k*-Center Clustering with Outliers in Sliding Windows. *Algorithms* **2022**, *15*, 52. https://doi.org/10.3390/a15020052

Academic Editor: Ruo-Wei Hung

Received: 11 January 2022
Accepted: 29 January 2022
Published: 31 January 2022

1. Introduction

In a number of modern scenarios (e.g., social network analysis, online finance, online transaction processing, etc.), data are produced as a continuous stream, and at such a high rate that on-the-fly processing can only afford to maintain a small portion of the data in memory, together with a limited amount of working space. This computational scenario is captured by the well-established *streaming model* [1]. The *sliding window setting* [2,3] introduces the additional desirable constraint that the input for the problem of interest consists of the window W of the most recent data items, whereas older data are considered "stale" and disregarded by the computation.

The *k*-center clustering problem (*k-center*, for short) is a fundamental unsupervised learning primitive with ubiquitous applications [4–6]. Given a set W of points from a metric space, the *k*-center problem requires determining a subset $C \subset W$ of k centers that minimize the maximum distance of any point of W from its closest center. However, since the objective function involves a maximum, the solution is at risk of being severely influenced by a few "distant" points, which are called *outliers*. In fact, the presence of outliers is inherent in many large datasets, since these points are often artifacts of data collection, either representing noisy measurements or simply erroneous information. To cope with this limitation, the *k*-center admits a heavily studied robust formulation that takes into account outliers [7–9]: when computing the objective function for a set of k centers, the z largest distances from the centers are to be discarded, where $z < |W|$ is an additional input parameter representing the tolerable level of noise. This formulation is known as the *k-center problem with z outliers*. We remark that the values of k and z are provided by the user and are typically chosen using either domain knowledge or some internal quality measure (see e.g., [10,11]).

The decision versions of the k-center problem and its variant with z outliers are NP-complete [12]; hence, only approximate solutions may be returned within reasonable time bounds. In this paper, we present approximation algorithms for the k-center problem with z outliers in the sliding window setting. Moreover, as a by-product, we also derive an algorithm to estimate the α-effective diameter of the window, which is an interesting measure of the spread of a noisy dataset.

1.1. Related Work

In the sequential setting, the k-center problem (without outliers) admits simple 2-approximation algorithms [12,13]. Recently, a sequential, randomized, fully dynamic (i.e., admitting arbitrary insertions and deletions of points) $(2+\varepsilon)$-approximation algorithm for the problem was presented in [14]. This algorithm features update times linear in k and $1/\varepsilon$, and polylogarithmic in the *aspect ratio* $\Delta = d_{\max}/d_{\min}$ of the pointset, that is, the ratio between the maximum distance $d_{\max}$ and minimum distance $d_{\min}$ of any two points in the set. For the k-center problem with z outliers, a simple, fully combinatorial 3-approximation algorithm running in $O(k|W|^2 \log(|W|))$ time was devised in [7]. In [8,9], an adaptation of this latter algorithm was proposed for weighted pointsets (where z represents the aggregate weight of the outliers). Recently, sequential 2-approximation algorithms were developed in [15,16] that, however, are based on a more complex LP-based approach that is less amenable to practical implementation. A bi-criteria randomized algorithm, which is only suitable for (small) constant k, that returns a 2-approximation as long as a slightly larger number of outliers is excluded from the objective function, was presented in [17].

For the classical streaming setting, which seeks, at any step, a solution for the *full* stream seen so far, a $(4+\varepsilon)$- approximation algorithm for the k-center problem with z outliers was given in [18]. The approximation was later improved to $3+\varepsilon$ in [9]. Whereas the former algorithm requires $O(kz/\varepsilon)$ working memory space, the latter requires space $O((k+z)(c/\varepsilon)^D)$, where c is a constant and D is the *doubling dimension* of the input stream, which is a widely used generalization of the notion of Euclidean dimension to arbitrary metrics [19,20], and is formally defined in Section 2.2.

For the stricter sliding window setting, in [21], the authors devised an algorithm able to compute a $(6+\varepsilon)$-approximation to the k-center problem for the current window, while keeping $O(k\log(\Delta)/\varepsilon)$ points stored in the working memory. Based on the same techniques, the authors also developed a $(3+\varepsilon)$-approximation algorithm for the diameter of the current window. In [22], we presented a $(2+\varepsilon)$-approximation algorithm for the k-center problem in the sliding window setting, where the improved approximation is obtained at the expense of a blow-up of a factor $O((c/\varepsilon)^D)$ in the working memory, with c constant.

Concerning the k-center problem with z outliers in the sliding window setting, the only known algorithm was devised very recently in [23]. At every time step, the algorithm maintains an ε-coreset for the problem on the current window, namely, a subset of the window points, such that, if used as an input of any c-approximation sequential algorithm for k-center with z outliers, yields a $(c+\varepsilon)$-approximate solution to the problem on the entire window. The algorithm requires the knowledge of the aspect ratio of the stream, and uses $O\big(\log(\Delta)kz(1/\varepsilon)^D\big)$ working memory. Moreover, since it relies on the execution of a sequential algorithm for k-center with z outliers on the coreset every time a new point is added, it has an $O\Big(\log(\Delta)\big(kz(1/\varepsilon)^D\big)^3\Big)$ update time per point, which makes it very impractical for streams with high arrival rates, and when values of k and z are not too small. In [23], the authors also prove that any sliding-window algorithm for k-center with z outliers, which features a $(1+\varepsilon)$ approximation ratio, must use $\Omega(\log(\Delta)kz/\varepsilon)$ working memory.

In [24,25], sliding window algorithms have been proposed for the k-median and k-means clustering problems, whose objective is to minimize the average distance and squared distance of all window points from the closest centers, respectively. For distributed solutions to the k-center problem (with and without outliers) targeting the volume rather than the velocity of the data, see [6,9] and the references therein.

The notion of the α-effective diameter was introduced in [26] in the context of graph analytics to characterize the growth rate of the neighborhood function, but it naturally extends to general metrics, providing a robust substitute of the diameter in the presence of noise. For $\alpha \in (0,1)$, the α-effective diameter of a metric dataset W is the minimum threshold such that the distances of at least $\alpha|W|^2$ pairs of window points fall below the threshold.

1.2. Our Contribution

We present approximation algorithms for the k-center with z outliers in the sliding window setting, which feature constant approximation ratios and small working memory requirements and update times. As is customary for clustering in the big data realm, our algorithms hinge on the maintenance of a *coreset*, that is, a small subset of representative points of the current window from which an accurate solution can be extracted. Crucial to the effectiveness of our approach is the introduction of *weights* for coreset points, where the weight of a point is (an estimate of) the number of window points it represents. To efficiently maintain the weights, we employ a succinct data structure inspired by the smooth histograms of [27], which enable considerable space savings if some slackness in the account of outliers is permitted. As an interesting by-product, we also devise an algorithm that employs our coresets to approximate the α-effective diameter of the current window W.

The analysis of our algorithms is carried out as a function of two design parameters, δ and λ, which control, respectively, the level of accuracy and the slackness in the account of outliers, and as a function of a number of characteristics of the stream S, namely, the values $d_{\min}$ and $d_{\max}$, representing the minimum and the maximum distance of two distinct points of S and the doubling dimension D of S.

Our main results are listed below.

- A sliding window algorithm for general metric spaces that, at any time, is able to return a set of centers covering all but at most $z(1+\lambda)$ points of the current window W, within a radius that is an $O(1)$ factor larger than the optimal radius for z ouliers. The algorithm requires a working memory of size $O\left(\log(d_{\max}/d_{\min})(k+z)\log_{1+\lambda}(|W|)\right)$ and processes each point in time linear in the working memory size. By setting $\lambda = 1/(2z)$, the number of uncovered points becomes, at most, z;
- An improved algorithm with the same coverage guarantee as above, featuring a radius that is only a factor $(3+O(\delta))$ larger than the optimal radius, at the expense of an extra $O((c/\delta)^D)$ factor in both the working memory size and update time, for a suitable constant c;
- A sliding-window algorithm for streams of bounded doubling dimension that, starting from a (possibly crude) lower bound on the ratio between the α-effective and the full diameter of the window W, returns upper and lower upper bounds to the α-effective diameter of W. The algorithm features accuracy–space tradeoffs akin to those of the improved algorithm for 1-center with $z = 0$ outliers;
- Experimental evidence that both the improved k-center and the effective diameter algorithms feature a good performance and provide accurate solutions.

It is important to remark that our algorithms are *fully oblivious* to the metric parameters $d_{\min}$, $d_{\max}$, and D, in the sense that the actual values of these parameters only influence the analysis but are not needed for the algorithms to run. This is a very desirable feature, since, in practice, these values are difficult to estimate.

Compared to the algorithm of [23] for k-center with z outliers, our algorithms feature a considerably lower update time, which makes them practically viable. Moreover, in the case of noisy streams for which $z = \Omega(\log|W|)$, our algorithms require considerably less working memory, as long as some slackness in the number of outliers can be tolerated. Finally, whereas the algorithm of [23] requires the knowledge of $d_{\min}$ and $d_{\max}$, our algorithms are oblivious to these values.

1.3. Organization of the Paper

The rest of the paper is structured as follows. Section 2 provides preliminary definitions. Sections 3 and 4 present, respectively, the algorithms for the k-center problem with z outliers and for the effective diameter. Section 5 reports on the experimental results. Section 6 concludes the paper with some final remarks and pointers to relevant open problems.

2. Preliminaries

Consider a (possibly unbounded) stream S of points from some metric space with distance function $\mathrm{dist}(\cdot,\cdot)$. At any time t, let W denote the set of the last $N = |W|$ points arrived, for a fixed window length N. In the sliding window model, for a given computational problem, we aim to develop algorithms that, at any time t, are able to solve the instance represented by the current window W, using working memory considerably smaller than N (possibly constant or logarithmic in N).

2.1. Definition of the Problems

For any point $p \in W$ and any subset $C \subseteq W$, we use the notation $\mathrm{dist}(p, C) = \min_{q \in C} \mathrm{dist}(p, q)$, and define the *radius of C with respect to W* as

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

For a positive integer $k < |W|$, the *k-center problem* requires finding a subset $C \subseteq W$ of k centers that minimizes $r_C(W)$. For a given W and k, we denote the radius of the optimal solution of this problem by $r^*_k(W)$. Given any radius value r, a subset $C \subseteq W$ with $r_C(W) \leq 2r$ can be incrementally built using the *greedy strategy* of [13]: starting from an arbitrary center, a new center selected among the points of W at a distance $> 2r$ from the current centers is iteratively added to C until all points of W are at a distance of at most $2r$ from C. An easy argument shows that, if $r \geq r^*_k(W)$, the set C obtained in this fashion has a size of, at most, k. By combining this strategy with a suitable guessing protocol, a 2-approximate solution to the k-center problem for W is obtained.

Note that any subset $C \subseteq W$ induces a partition of W into $|C|$ clusters by assigning each point to its closest center (with ties broken arbitrarily).

In this paper, we focus on the following important extension to the k-center problem. For positive $k, z < |W|$, the *k-center problem with z outliers* requires finding a subset $C \subseteq W$ of size k minimizing $r_C(W - Z_C)$, where Z_C is the set of z points in W with the largest distances from C, which are regarded as outliers to be discarded from the clustering. We denote the radius of the optimal solution of this problem by $r^*_{k,z}(W)$. Observe that the k-center problem with z outliers reduces to the k-center problem for $z = 0$. In addition, it is straightforward to argue that the optimal solution of the k-center problem (without outliers) with $k + z$ centers has a radius not larger than the optimal solution of the problem with k centers and z outliers, that is,

$$r^*_{k+z}(W) \leq r^*_{k,z}(W). \tag{1}$$

In a more general formulation of the k-center problem with z outliers, each point $p \in W$ carries a positive integer weight $w(p)$, and the desired set C of k centers must minimize $r_C(W - Z_C)$, where Z_C is the set of points with the largest distances from C of maximum cardinality and an aggregate weight of, at most, z. We will refer to this weighted formulation as *k-center with z weighted outliers*.

The algorithms presented in this paper for k-center with z outliers crucially rely on the extraction of a succinct *coreset* T from the (possibly large) input W, so a solution to the problem can be efficiently computed by running a sequential algorithm on T rather than on W. The quality of a coreset T is captured by the following 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 with z outliers if* $\max_{p \in W} \mathrm{dist}(p, T) \leq \varepsilon r^*_{k,z}(W)$.

An ε-coreset T of W ensures that each point in W is "represented" by a close enough point in T, where closeness is defined w.r.t. ε and $r^*_{k,z}(W)$. In fact, our algorithms will make use of *weighted coresets*, where, additionally, each coreset point $p \in T$ features a weight that is (an approximation of) the number of points of W represented by p.

An important characteristic of a pointset W is its *diameter*, defined as $\Delta_W = \max_{p,q \in W} \mathrm{dist}(p, q)$, which can be computed exactly in quadratic time. The diameter is very sensitive to noise in the dataset and, in the presence of outliers, its value might turn out to be scarcely representative of most pairwise distances in W. Thus, the more robust notion of the *effective diameter* has been introduced in [26]. Let $d_{1,W}, d_{2,W}, \ldots$ be an enumeration of the $|W|^2$ distances between all pairs of points of W in non-decreasing order. For a given parameter $\alpha \in (0, 1)$, the *α-effective diameter* of W is defined as $\Delta^\alpha_W = d_{\lceil \alpha |W|^2 \rceil, W}$, namely, the smallest value such that at least $\alpha|W|^2$ pairs of points in W are within distance Δ^α_W.

2.2. Doubling Dimension

The analysis of our algorithms will be carried out as a function of a number of relevant parameters, including the dimensionality of the data. To deal with arbitrary metric spaces, we resort to the following well-established general notion of dimensionality. For any $x \in W$ and $r > 0$, the *ball of radius r centered at x*, denoted as $B(x, r)$, is the subset of all points of W at a distance of, at most, r from x. The *doubling dimension* of W is the minimum value D such that, for all $x \in W$, any ball $B(x, r)$ is contained in the union of, at most, 2^D balls of radius $r/2$ centered at points of W. The notion of doubling dimension has been used extensively in previous works (see [22,28] and the references therein).

3. k-Center with z Outliers

Let S be a (possibly unbounded) stream of points from some metric space, and let N be the selected window length. For any point $p \in S$, its *Time-To-Live* $\mathrm{TTL}(p)$ is N when p arrives, and it decreases by 1 at each subsequent step. We say that p is *active* when $\mathrm{TTL}(p) > 0$, and that it *expires* when $\mathrm{TTL}(p)$ becomes 0. For convenience, the analysis will also consider expired points with negative TTLs. At any time t, the current window W consists of all arrived points with positive TTL; hence, $|W| = N$.

In this section, we present coreset-based algorithms that, at any time t, are able to return accurate approximate solutions to the k-center problem with z outliers for W. The section is structured as follows. Section 3.1 describes and analyzes the weighted coreset construction. Section 3.2 discusses how to extract the final solution from the weighted coreset whose radius is, at most, a constant factor away from $r^*_{k,z}(W)$, as long as a slightly larger number of outliers is tolerated. Section 3.3 shows how to remove an assumption made to simplify the presentation. Finally, Section 3.4 shows that, for spaces of bounded doubling dimension, the approximation factor can be lowered to a mere $3 + \varepsilon$ for any fixed ε at the expense of larger working memory requirements.

3.1. Weighted Coreset Construction

3.1.1. Algorithm

The proposed coreset construction hinges upon the approach by [21] for k-center without outliers, with major extensions introduced to maintain weights. Let $d_{\min}$ and $d_{\max}$ denote, respectively, the minimum and maximum distances between any two distinct points of the stream. For a user-defined constant $\beta \in (0, 1]$, let

$$\Gamma = \{(1+\beta)^i : \lfloor \log_{1+\beta} d_{\min} \rfloor \leq i \leq \lceil \log_{1+\beta} d_{\max} \rceil\},$$

The values in Γ will be used as guesses of the optimal radius $r^*_{k+z}(W)$ of a $(k+z)$-center clustering without outliers of the current window (recall that $r^*_{k+z}(W)$ is a lower bound to the optimal radius $r^*_{k,z}(W)$ for the problem with z outliers), and the algorithm will maintain suitable data structures capable of identifying the right guess. For ease of presentation, we assume for now that $d_{\min}$ and $d_{\max}$ are known to the algorithm. In Section 3.3, we will show how the assumption can be removed by maintaining estimates of the two values. For each guess γ, the algorithm maintains three sets of active points, namely A_γ, R_γ, and O_γ, and the coreset is extracted from these sets. A_γ is a small set of active points, called *attraction points*, such that, for any two distinct $a_1, a_2 \in A_\gamma$, $\text{dist}(a_1, a_2) > 2\gamma$.

At every time t, the arrival of a new point p is handled as follows, for every guess γ. If there exist attraction points a such that $\text{dist}(p, a) \le 2\gamma$, we define $a_\gamma(p)$ as the one with minimum TTL, and say that p *is attracted by* $a_\gamma(p)$. Otherwise, p becomes a new attraction point in A_γ, and we let $a_\gamma(p) = p$ (i.e., p is attracted by itself). Set R_γ maintains, for each $a \in A_\gamma$, one *representative* $r_\gamma(a)$, defined as the most recent point attracted by a. Note that, whereas $a_\gamma(p)$ is fixed at p's arrival, the representative $r_\gamma(a)$ may change with time. When an attraction point a expires, its representative $r_\gamma(a)$ becomes an *orphan* and is moved to the set O_γ.

Since the pairwise distance between the points of A_γ is $> 2\gamma$, if $|A_\gamma| \ge k+z+1$, it is clear that $\gamma < r^*_{k+z}(W)$, and, as will be seen below, the points in $A_\gamma \cup R_\gamma \cup O_\gamma$ will not be used to extract the coreset. Therefore, to save memory, we set $k+z+1$ as a threshold for $|A_\gamma|$: when $|A_\gamma| = k+z+1$ and the newly arrived point qualifies to be an attraction point, the algorithm discards the point $a \in A_\gamma$ with minimum TTL, and moves its representative $r_\gamma(a)$ to O_γ. As further space saving, all points in O_γ older than a are discarded, since throughout their residual lifespan, $|A_\gamma| \ge k+z+1$, and hence they cannot contribute to a valid coreset.

At any time t, the coreset for the k-center problem with z outliers, w.r.t. the current window W, is obtained as $T = R_{\hat\gamma} \cup O_{\hat\gamma}$, where $\hat\gamma$ is the smallest guess such that: (i) $|A_{\hat\gamma}| \le k+z$; and (ii) by running the simple greedy strategy of [13], reviewed in Section 2.1, a set C of $k+z$ points can be selected from $A_{\hat\gamma} \cup R_{\hat\gamma} \cup O_{\hat\gamma}$, such that any other point in this set is at a distance of, at most, $2\hat\gamma$ from a selected point. To ensure that an accurate solution to the k-center problem with z outliers can be extracted from the coreset T, we need to weigh each point $p \in T$ with (a suitable accurate estimate of) the number of window points for which p can act as a *proxy*. This requires maintaining additional information with the points of the various sets R_γ and O_γ, as explained below.

For each guess γ and each active point $p \in W$, we define its proxy $\pi_\gamma(p) \in R_\gamma \cup O_\gamma$ as the most recent active point r such that both p and r are attracted by $a_\gamma(p)$. (Note that r may be an orphan if $a_\gamma(p)$ was discarded from A_γ.) Thus, $\pi_\gamma(p) = r_\gamma(a_\gamma(p))$. Therefore, at any time t, the proxy function $\pi_\gamma(\cdot)$ defines a mapping between active points and points of $O_\gamma \cup R_\gamma$, and, for every $r \in O_\gamma \cup R_\gamma$ we define its *weight* $w_\gamma(r) = |\{p \in W \ : \ \pi_\gamma(p) = r\}|$. For each $r \in R_\gamma \cup O_\gamma$, our algorithm maintains a *histogram* $L_{r,\gamma} = \{(t_{r,1}, c_{r,1}), (t_{r,2}, c_{r,2}), \ldots\}$, which is a list of pairs (timestamp, weight) such that there are $c_{r,j}$ points p assigned to r (i.e., for which $\pi_\gamma(p) = r$) that arrived at or after time $t_{r,j}$. When a point p arrives at time t, the histograms are updated as follows (for the sake of readability, from now on, we drop the subscript γ from histograms and weights, when clear from the context.). If p becomes a new attraction point (hence, $p = a_\gamma(p) = r_\gamma(p)$), a new histogram $L_p = \{(t, 1)\}$ is created and is assigned to p. If, instead, p is attracted by some $a \in A_\gamma$, then p becomes the representative $r_\gamma(a)$ and inherits the histogram from the previous representative, modified by increasing all weights by 1 and adding the new pair $(t, 1)$. In addition, all histogram entries with timestamp $t - |W|$ are discarded, since they refer to the point that expired at time t. The pairs in each histogram are naturally sorted by an increasing order of timestamps and decreasing order of weight. Observe that, when a representative r becomes an orphan, and hence is moved from R_γ to O_γ, its histogram L_r does not acquire new entries until r expires.

At any time t, for a histogram L_r, we denote by c_{L_r} the weight of the pair in L_r with the smallest timestamp (which is greater than or equal to $t - |W| + 1$ by virtue of the elimination of the old entries described above). It is easy to see that $c_{L_r} = w(r)$, that is, the number of points p for which r is the proxy. Unfortunately, keeping the full histogram L_r for each $r \in R_\gamma \cup O_\gamma$ requires a working memory of size $\Theta(|W|)$, which is far beyond the space bound targeted by sliding window algorithms. Therefore, taking inspiration from the smooth histograms of [27], we maintain in L_r only a trimmed version of the full list, which, however, ensures that c_{L_r} is an estimate of $w(r)$ with a controlled level of accuracy. Specifically, let $\lambda > 0$ be a user-defined accuracy parameter. Every time a histogram L_r is updated, a scan of the pairs is performed that implements the following trimming:

- The first pair (t, c) is kept in the histogram;
- If a pair (t, c) is kept in the histogram, all subsequent pairs (t', c') with $t' > t$ and $c \le (1+\lambda)c'$ are deleted, except for the last such pair, if any.

At any time t, the weighted coreset that will be used to solve the k-center problem with z outliers for the current window W consists of the set $T = R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$, where the guess $\hat{\gamma}$ is computed as described above, and each $r \in T$ is assigned weight $\tilde{w}(r) = c_{L_r}$. As will be shown in the next subsection, the $\tilde{w}(r)$s are good approximations of the true weights $w(r)$s, and this will provide good bi-criteria approximation quality for the returned solution.

The pseudocode detailing the algorithm is provided below. The arrival of a new point p at time t is handled by the main Procedure UPDATE(p, t) (Algorithm 1), which, for every guess γ, invokes, in turn, Procedure INSERTATTRACTION(p, γ) (Algorithm 2) when p must be added to A_γ, or Procedure UPDATEHISTROGRAMS$(L_{p,\gamma})$ (Algorithm 3) when p becomes a new representative of some existing point of A_γ. To extract the weighted coreset, Procedure EXTRACTCORESET() (Algorithm 4) is executed.

Algorithm 1: UPDATE(p, t)

```
1  foreach $\gamma \in \Gamma$ do
2      foreach expired $p \in A_\gamma$ do
3          $A_\gamma \leftarrow A_\gamma \setminus \{p\}$
4          Move $r_\gamma(p)$ from $R_\gamma$ to $O_\gamma$
5      foreach $r \in O_\gamma$ do
6          if $r$ is expired then
7              Remove $r$ (and its histogram) from $O_\gamma$
8          Remove from $L_{r,\gamma}$ the entry (if any) with timestamp $t - |W|$
9      $x \leftarrow \text{argmin}_{q \in A_\gamma : \text{dist}(p,q) \le 2\gamma} \text{TTL}(q)$
10     if $x == null$ then
11         INSERTATTRACTION$(p, \gamma)$
12         $L_{p,\gamma} \leftarrow \{(t, 1)\}$
13     else
14         Move the content of $L_{r_\gamma(x),\gamma}$ to $L_{p,\gamma}$
15         UPDATEHISTOGRAM$(L_{p,\gamma})$
16         Set $r_\gamma(x) = p$ in $R_\gamma$
```

Algorithm 2: INSERTATTRACTION(p, γ)

```
1  $A_\gamma \leftarrow A_\gamma \cup \{p\}$
2  $r_\gamma(p) \leftarrow p$
3  $R_\gamma \leftarrow R_\gamma \cup \{r_\gamma(p)\}$
4  if $|A_\gamma| > k + z + 1$ then
5      $v_{old} \leftarrow \text{argmin}_{v \in A_\gamma} \text{TTL}(v)$
6      $A_\gamma \leftarrow A_\gamma \setminus \{v_{old}\}$
7      Move $r_\gamma(v_{old})$ from $R_\gamma$ to $O_\gamma$
8  if $|A_\gamma| > k + z$ then
9      $t_{\min} \leftarrow \min_{v \in A_\gamma} \text{TTL}(v)$
10     Remove from $O_\gamma$ all $q$ with $\text{TTL}(q) < t_{\min}$ (and their histograms)
```

Algorithm 3: UPDATEHISTOGRAM(L)

```
Let $L[i] = (L[i].t, L[i].c)$ denote the $i$th pair in $L$, for $i = 1, 2, \ldots |L|$
for $i = 1$ to $|L|$ do
    $L[i].c \leftarrow L[i].c + 1$
Append $(t, 1)$ to $L$
Create a new histogram $M = \{L[1]\}$
$last = 1$
for $i = 2$ to $|L| - 1$ do
    if $L[last].c > (1 + \lambda)L[i + 1].c$ then
        Append $L[i]$ to $M$
        $last = i$
Append $L[|L|]$ to $M$
$L = M$
```

Algorithm 4: EXTRACTCORESET()

```
for increasing $\gamma \in \Gamma$ such that $|A_\gamma| \leq k + z$ do
    $C \leftarrow \emptyset$
    for $p \in A_\gamma \cup O_\gamma \cup R_\gamma$ do
        if $\mathrm{dist}(p, C) > 2\gamma$ then $C \leftarrow C \cup \{p\}$
    if $|C| \leq k + z$ then
        $\hat{\gamma} \leftarrow \gamma$
        break;
$T \leftarrow R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$
foreach $r \in T$ do $\tilde{w}(r) = c_{L_r}$
return $T$ together with the approximate weights.
```

3.1.2. Analysis

The following two technical lemmas state important properties of the sets A_γ, O_γ, and R_γ, and of the histograms maintained by the algorithm.

Lemma 1. *At any time t, the following properties hold for every $\gamma \in \Gamma$:*

1. *If $|A_\gamma| \leq k + z$, then $\max_{q \in W} \mathrm{dist}(q, \pi_\gamma(q)) \leq 4\gamma$.*
2. *$|A_\gamma|, |R_\gamma|, |O_\gamma| \leq k + z + 1$.*

Proof. In order to prove the lemma, it suffices to show that, if the properties hold after the processing of the $(t-1)$-th point, they are inductively maintained after the invocation of UPDATE(p, t). The proof makes use of essentially the same arguments employed in [21] [Lemmas 7, 8], straightforwardly adapted to account for the fact that, in our algorithm, each new point that does not become an attraction point is made representative of a single attraction point, whereas, in [21], it would be made representative of all attraction points at a distance of, at most, 2γ. □

Recall that, at any time t and for any guess γ, the histogram L_r associated with each point $r \in R_\gamma \cup O_\gamma$ is a list of pairs $(t_{r,i}, c_{r,i})$ indicating that there are currently $c_{r,i}$ points that have arrived after time $\geq t_{r,i}$, whose proxy is r, for $i = 1, 2, \ldots$. We have:

Lemma 2. *For any time t, guess γ, and $r \in R_\gamma \cup O_\gamma$, the following properties hold for L_r.*

1. *For every $1 \leq i \leq |L_r|$, $c_{r,i} \leq |W|$;*
2. *For every $1 \leq i \leq |L_r| - 1$, $c_{r,i} \leq (1 + \lambda)c_{r,i+1}$ or $c_{r,i} = 1 + c_{r,i+1} > (1 + \lambda)c_{r,i+1}$;*
3. *For every $1 \leq i \leq |L_r| - 2$, $c_{r,i} > (1 + \lambda)c_{r,i+2}$;*
4. *$|L_r| \in O(\log_{1+\lambda} |W|)$.*

Proof. First, observe that Property 4 is an immediate consequence of Properties 1 and 3. Hence, we are left with proving Properties 1, 2, and 3. The properties clearly hold when the histogram L_r is first created (Line 12 of UPDATE); thus, we only need to show that if they hold prior to an invocation of UPDATEHISTOGRAM(L_r), they continue to hold for the histogram M created by UPDATEHISTOGRAM(L_r), which becomes the new L_r at the end of the procedure. Property 1 holds since the weights are updated only as long as $a_\gamma(r)$, the oldest point accounted for in the histogram, is active; hence, weights always represent sizes of subsets of points of the same window, and thus they never exceed $|W|$. Consider now Properties 2 and 3. Let L'_r denote the histogram L_r after the execution of Line 4 of UPDATEHISTOGRAM(L_r) (i.e., after the increment of the weights and the addition of $(t, 1)$ at the end of the list). It is easy to argue that Properties 2 and 3 continue to hold for L'_r. Let us now show that they also hold for histogram M at the end of the procedure. As for Property 2, consider two adjacent pairs $(t_{r,i}, c_{r,i})$ and $(t_{r,i+1}, c_{r,i+1})$ in M, with $c_{r,i} > (1+\lambda)c_{r,i+1}$. Then, two pairs with the same timestamps and weights must exist in L'_r, and we can argue that these two pairs must be adjacent in L'_r, and hence their weights must differ by 1, since Property 2 holds for L'_r. Indeed, if the two pairs were not adjacent in L'_r, then at least one pair with a timestamp between $t_{r,i}$ and $t_{r,i+1}$ must have been removed in the for loop of UPDATEHISTOGRAM(L_r). However, this is not possible, because, due to the way the loop operates, this would ensure that $c_{r,i} \le (1+\lambda)c_{r,i+1}$. Finally, Property 3 is enforced by the for loop of UPDATEHISTOGRAM(L_r). □

The following theorem states the main properties of the weighted coreset T computed by our algorithm.

Theorem 1. *At any time t, the weighted coreset T returned by* EXTRACTCORESET() *is a $4(1+\beta)$-coreset of size $O(k+z)$ for the current window W w.r.t. the k-center problem with z outliers. Moreover, for each point $r \in T$, we have $w(r)/(1+\lambda) \le \tilde{w}(r) \le w(r)$.*

Proof. Recall that $T = R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$, where $\hat{\gamma}$ is the minimum guess for which the following two conditions are verified: $|A_{\hat{\gamma}}| \le k+z$, and the greedy selection strategy of [13] applied to $A_{\hat{\gamma}} \cup R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ returns a set C of $k+z$ points such that $\text{dist}(p, C) \le 2\hat{\gamma}$, for every $p \in A_{\hat{\gamma}} \cup R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$. It is easy to see that these two conditions are surely verified for any $\gamma \ge r^*_{k+z}(W)$. Therefore, considering the density of the guesses in Γ, we must have $\hat{\gamma} \le (1+\beta)r^*_{k+z}(W)$. By Lemma 1, we have $|T| = O(k+z)$ and $\max_{p\in W} \text{dist}(p, T) \le 4\hat{\gamma}$, which implies $\max_{p\in W} \text{dist}(p, T) \le 4(1+\beta)r^*_{k+z}(W) \le 4(1+\beta)r^*_{k,z}(W)$.

We now show the relation concerning the approximate weights $\tilde{w}(r)$. Recall that, for each $r \in T$, the value $\tilde{w}(r)$ is set equal to c_{L_r}, which is the weight component of the pair in L_r with the smallest timestamp $\ge t - |W| + 1$. Let $(t_{r,i}, c_{r,i})$ be the pair of L_r such that $c_{r,i} = c_{L_r}$. If $i > 1$, then the relation $c_{r,i} \le w(r) \le c_{r,i-1}$ must hold, and Property 2 of Lemma 2 ensures that $c_{r,i-1} \le c_{r,i}(1+\lambda)$ or $c_{r,i-1} = c_{r,i} + 1$. In the former case, we have $w(r)/(1+\lambda) \le c_{r,i} = c_{L_r} \le w(r)$, whereas, in the latter case, it is easy to see that $c_{L_r} = w(r)$. If, instead, $i = 1$, it is easy to see that $c_{L_r} = w(r)$, since the first pair of the histogram is always relative to the arrival of the attraction point $a_\gamma(r)$, and hence the weight of such pair accounts for all points assigned to it. □

The following theorem analyzes the space and time performance of our coreset construction strategy.

Theorem 2. *The data structures used by our coreset construction strategy require a working memory of size $O\Big(\log_{1+\beta}(d_{\max}/d_{\min}) \cdot (k+z) \cdot \log_{1+\lambda}(|W|)\Big)$. Moreover, Procedure* UPDATE *can be implemented to run in time*

$$O\Big(\log_{1+\beta}(d_{\max}/d_{\min}) \cdot \big((k+z) + \log_{1+\lambda}(|W|)\big)\Big),$$

whereas Procedure EXTRACTCORESET *can be implemented to run in time*

$$O\Big(\log_{1+\beta}(d_{\max}/d_{\min}) \cdot (k+z)^2\Big).$$

If binary search is used to find the value $\hat{\gamma}$, the running time of EXTRACTCORESET *decreases to*

$$O\Big(\log(\log_{1+\beta}(d_{\max}/d_{\min}))\cdot(k+z)^2\Big).$$

Proof. The bound on the working memory is an immediate consequence of Lemmas 1 and 2, and of the fact that $|\Gamma| = O\Big(\log_{1+\beta}(d_{\max}/d_{\min})\Big)$. Procedure UPDATE is easily implemented through a constant number of linear scans of A_γ, R_γ, and O_γ, for every $\gamma \in \Gamma$, and a linear scan of, at most, one histogram. For what concerns Procedure EXTRACTCORESET, for each $\gamma \in \Gamma$, the computation of C requires time quadratic in $|A_\gamma| + |R_\gamma| + |O_\gamma|$, and the number of guesses $\gamma \in \Gamma$ to be checked are at most $|\Gamma|$ if a linear search is used, and $O(\log(|\Gamma|))$ if binary search is used. In addition, time $O(|T|)$ is needed to compute $\tilde{w}(r)$ for all $r \in T$. Based on these considerations, the complexity bounds follow again from Lemmas 1 and 2 and the size of $|\Gamma|$. □

3.2. Computation of the Solution from the Coreset

At any time t, to compute a solution to the k-center problem with z outliers on the window W, we first determine a weighted coreset T, as explained in the previous subsection, and then, on T, we run a sequential strategy for the weighted variant of the problem. To this purpose, we make use of the algorithm developed in [8,9], which generalizes the sequential algorithm of [7] to the weighted case. Given as an input a weighted set T, a value k, a precision parameter ε, and a radius ρ, the algorithm computes a set X of k centers incrementally in, at most, k iterations. Initially, all points are considered *uncovered*. At each iteration, the next center is selected as the point that maximizes the aggregate weight of the yet uncovered points within distance $(1+2\varepsilon)\rho$, and such a center *covers* all points in T within the larger distance $(3+4\varepsilon)\rho$. The algorithm terminates when either $|X| = k$ or no uncovered points remain, returning X and the subset T' of uncovered points. This algorithm is implemented by procedure OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$. The next lemma states an important property related to the use of OUTLIERSCLUSTER (Algorithm 5) in our context.

Algorithm 5: OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$

```
1  $T' = T$
2  $X = \emptyset$
3  while $|X| < k$ and $T' \neq \emptyset$ do
4      for $r \in T$ do
5          $B_r = \{v : v \in T' \text{and } \mathrm{dist}(v,r) \le (1+2\varepsilon)\rho\}$
6      $x = \mathrm{argmax}_{r\in T} \sum_{v\in B_r} \tilde{w}(v)$
7      $X = X \cup \{x\}$
8      $E_x = \{v : v \in T' \text{and } \mathrm{dist}(v,x) \le (3+4\varepsilon)\rho\}$
9      $T' = T' \setminus E_x$
10 return $X, T'$
```

Lemma 3. *For any time t, let $T = R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ be the weighted coreset returned by* EXTRACTCORESET(). *For any $\rho \ge r^*_{k,z}(W)$, the invocation* OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$ *with $\varepsilon = 4(1+\beta)$ returns a set of centers X and a set of uncovered points $T' \subseteq T$, such that*

- $\mathrm{dist}(r, X) \le (3+4\varepsilon)\rho \quad \forall r \in T \setminus T'$
- $\sum_{r\in T'} \tilde{w}(r) \le z$.

Proof. The proof can be obtained as a simple technical adaptation of the one of [9] [Lemma 5] to the case of approximate weights. For clarity, we detail the entire proof rather than highlighting the changes only. The bound on $\mathrm{dist}(r, X)$ for every $r \in T \setminus T'$ is directly enforced by the algorithm. We are left to show that $\sum_{r\in T'} \tilde{w}(r) \le z$. If $|X| < k$, then $T' = \emptyset$ and the claim holds trivially. We now concentrate on the case $|X| = k$. Consider the i-th iteration of the while loop of OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$ and define x_i as the center of X selected in the iteration, and T'_i as the set T' of uncovered points at the beginning of the iteration. Recall that x_i is the point of T that maximizes the cumulative approximate weight of the set B_{x_i} of uncovered points in T'_i at a distance of, at most, $(1+2\varepsilon)\cdot\rho$ from x_i, and that the set E_{x_i} of all uncovered points at a distance of, at most, $(3+4\varepsilon)\cdot\rho$ from x_i is removed from T'_i at the end of the iteration. Let

$$\sigma_T = \sum_{r\in T} \tilde{w}(r).$$

We now show that

$$\sum_{i=1}^{k} \sum_{r\in E_{x_i}} \tilde{w}(r) \geq \sigma_T - z, \tag{2}$$

which will immediately imply that $\sum_{r\in T'} \tilde{w}(r) \leq z$. To this purpose, let O be an optimal set of k centers for the current window W, and let Z be the set of, at most, z outliers at a distance greater than $r^*_{k,z}(W)$ from O. Let also $\tilde{W}$ be a subset of the current window that, for every $r \in T$, contains exactly $\tilde{w}(r)$ points q, including r, for which r is a proxy, that is, points $q \in W$ such that $\pi_{\hat{\gamma}}(q) = r$. Note that $\tilde{W}$ is well defined since, by Theorem 1, $\tilde{w}(r)$ is always less than or equal to the actual weight of r. For each $o \in O$, define $C_o \subseteq \tilde{W} \setminus Z$ as the set of nonoutlier points in $\tilde{W}$ that are closer to o than to any other center of O, with ties broken arbitrarily. It is important to remark, that while there may be some optimal center $o \in O$ that is not in $\tilde{W}$, its proxy $\pi_{\hat{\gamma}}(o)$ is in T; hence, it is guaranteed to be in $\tilde{W}$. To prove Equation (2), it is sufficient to exhibit an ordering $o_1, o_2, \ldots, o_k$ of the centers in O so that, for every $1 \leq i \leq k$, it holds

$$\sum_{j=1}^{i} \sum_{r\in E_{x_j}} \tilde{w}(r) \geq |C_{o_1} \cup \cdots \cup C_{o_i}|.$$

The proof uses an inductive charging argument to assign each point in $\bigcup_{j=1}^{i} C_{o_j}$ to a point in $\bigcup_{j=1}^{i} E_{x_j}$, where each r in the latter set will be in charge of, at most, $\tilde{w}(r)$ points. We define two charging rules. A point can be either charged to its own proxy (*Rule 1*) or to another point of T (*Rule 2*).

Fix some arbitrary i, with $1 \leq i \leq k$, and assume, inductively, that the points in $C_{o_1} \cup \cdots \cup C_{o_{i-1}}$ have been charged to points in $\bigcup_{j=1}^{i-1} E_j$ for some choice of distinct optimal centers $o_1, o_2, \ldots, o_{i-1}$. We have two cases.

Case 1. There exists an optimal center o still unchosen such that there is a point $v \in C_o$ with $\pi_{\hat{\gamma}}(v) \in B_{x_j}$, for some $1 \leq j \leq i$. We choose o_i as one such center. Hence, $d(x_j, \pi_{\hat{\gamma}}(v)) \leq (1+2\varepsilon)\cdot\rho$. By repeatedly applying the triangle inequality, we have that, for each $u \in C_{o_i}$,

$$\begin{aligned} d(x_j, \pi_{\hat{\gamma}}(u)) \leq{} & d(x_j, \pi_{\hat{\gamma}}(v)) + d(\pi_{\hat{\gamma}}(v), v) + d(v, o_i) \\ & + d(o_i, u) + d(u, \pi_{\hat{\gamma}}(u)) \leq (3+4\varepsilon)\cdot\rho \end{aligned}$$

hence, $\pi_{\hat{\gamma}}(u) \in E_{x_j}$. Therefore we can charge each point $u \in C_{o_i}$ to its proxy by Rule 1.

Case 2. For each unchosen optimal center o and each $v \in C_o$, $\pi_{\hat{\gamma}}(v) \notin \bigcup_{j=1}^{i} B_{x_j}$. We choose o_i to be the unchosen optimal center that maximizes the cardinality of $\{\pi_{\hat{\gamma}}(u) : u \in C_{o_i}\} \cap T'_i$. We distinguish between points $u \in C_{o_i}$ with $\pi_{\hat{\gamma}}(u) \notin T'_i$; hence, $\pi_{\hat{\gamma}}(u) \in \bigcup_{j=1}^{i-1} E_{x_j}$, and those with $\pi_{\hat{\gamma}}(u) \in T'_i$. We charge each $u \in C_{o_i}$ with $\pi_{\hat{\gamma}}(u) \notin T'_i$ to its own proxy by Rule 1. As for the other points, we now show that we can charge them to the points of

B_{x_i}. To this purpose, we first observe that $B_{\pi_{\hat{\gamma}}(o_i)}$ contains $\{\pi_{\hat{\gamma}}(u) : u \in C_{o_i}\} \cap T'_i$, since, for each $u \in C_{o_i}$,

$$\begin{aligned} d(\pi_{\hat{\gamma}}(o_i), \pi_{\hat{\gamma}}(u)) &\le d(\pi_{\hat{\gamma}}(o_i), o_i) + d(o_i, u) + d(u, \pi_{\hat{\gamma}}(u)) \\ &\le (1+2\varepsilon) \cdot r^*_{k,z}(W) \le (1+2\varepsilon) \cdot \rho. \end{aligned}$$

Therefore, the aggregate approximate weight of $B_{\pi_{\hat{\gamma}}(o_i)}$ is at least $\left|\{u \in C_{o_i} : \pi_{\hat{\gamma}}(u) \in T'_i\}\right|$. Since Iteration i selects x_i as the center, such that B_{x_i} has maximum aggregate approximate weight, we have that

$$\sum_{r \in B_{x_i}} \tilde{w}(r) \ge \sum_{r \in B_{\pi_{\hat{\gamma}}(o_i)}} \tilde{w}(r) \ge \left|\{u \in C_{o_i} : \pi_{\hat{\gamma}}(u) \in T'_i\}\right|,$$

hence, the points $u \in C_{o_i}$ with $\pi_{\hat{\gamma}}(u) \in T'_i$ can be charged to the points in B_{x_i}, since B_{x_i} has enough aggregate approximate weight.

Note that the points of B_{x_i} did not receive any charging by Rule 1 in previous iterations, since they are uncovered at the beginning of Iteration i, and will not receive chargings by Rule 1 in subsequent iterations, since B_{x_i} does not intersect the set C_o of any optimal center o yet to be chosen. In addition, no further charging to the points of B_{x_i} by Rule 2 will happen in subsequent iterations, since Rule 2 will only target sets B_{x_h} with $h > i$. These observations ensure that any point of T receives charges through either Rule 1 or Rule 2, but not both, and never in excess of its weight, and the proof follows. □

At any time t, to obtain the desired solution, we invoke Procedure COMPUTESOLUTION, which works as follows (see Algorithm 6 for the pseudocode). The procedure first extracts the coreset $T = R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ calling EXTRACTCORESET. Then, it sets $\varepsilon = 4(1+\beta)$ and runs OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$ for a geometric sequence of values of ρ between $d_{\min}$ and $d_{\max}$, with step $1+\beta$, stopping at the minimum value ρ_{min} for which the pair (X, T') returned by OUTLIERSCLUSTER$(T, k, \rho_{min}, \varepsilon)$ is such that the aggregate approximate weight of set T' is, at most, z. (Note that the parameter β in the definitions of ε and of the step used in the geometric search for ρ_{min} is the same one that appears in the definition of Γ; hence, $\beta \in (0, 1]$.) At this point, the set of centers X is returned as a solution to the k-center problem with z outliers on the current window W.

Algorithm 6: COMPUTESOLUTION()

```
1  $T \leftarrow$ EXTRACTCORESET()
2  $\rho \leftarrow d_{\min}$
3  $\varepsilon \leftarrow 4(1+\beta)$
4  $(X, T') \leftarrow$ OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$
5  $\tilde{w}(T') \leftarrow \sum_{r \in T'} \tilde{w}(r)$
6  while $\tilde{w}(T') > z$ do
7      $\rho \leftarrow \rho \cdot (1+\beta)$
8      $(X, T') \leftarrow$ OUTLIERSCLUSTER$(T, k, \rho, \varepsilon)$
9      $\tilde{w}(T') \leftarrow \sum_{r \in T'} \tilde{w}(r)$
10 return $X$
```

The following theorem highlights the tradeoff between the accuracy (in terms of both the radius and excess number of outliers) and the performance exhibited by COMPUTESOLUTION.

Theorem 3. *At any time t, COMPUTESOLUTION returns a set $X \subseteq W$ of, at most, k centers, such that at least $|W| - (1+\lambda)z$ points of W are at a distance of, at most, $(23+55\beta)r^*_{k,z}(W)$ from X. The procedure requires a working memory of size $O\Big((k+z)\log_{1+\beta}(d_{\max}/d_{\min})\log_{1+\lambda}(|W|)\Big)$ and runs in time*

$$O\Big(\log_{1+\beta}(d_{\max}/d_{\min}) \cdot k(k+z)^2\Big).$$

If the while loop is substituted by a binary search for ρ_{min}, the running time decreases to

$$O\Big(\log(\log_{1+\beta}(d_{\max}/d_{\min}))\cdot k(k+z)^2\Big).$$

Proof. Let ρ_{min} be such that OUTLIERSCLUSTER$(T,k,\rho_{min},\varepsilon)$ yields (X,T'), where X is the returned solution, and let W' be the set of points of W whose proxies are in T'. By Lemma 3 and the choice of the step of the geometric search for ρ_{min}, we have that $\rho_{min}\le(1+\beta)r^*_{k,z}(W)$; hence, for each $p\in W-W'$, $\text{dist}(\pi_{\hat{\gamma}}(p),X)\le(3+4\varepsilon)\rho_{min}\le(3+4\varepsilon)(1+\beta)r^*_{k,z}(W)$. Since T is an ε-coreset (Theorem 1) and $\beta\in(0,1]$, we conclude that, for every $p\in W-W'$,

$$\begin{aligned}\text{dist}(p,X) &\le \text{dist}(p,\pi_{\hat{\gamma}}(p))+\text{dist}(\pi_{\hat{\gamma}}(p),X)\\ &\le \varepsilon r^*_{k,z}(W)+(3+4\varepsilon)(1+\beta)r^*_{k,z}(W)\\ &< (23+55\beta)r^*_{k,z}(W)\end{aligned}$$

Moreover, by Theorem 1, we also have that $|W'|\le(1+\lambda)\sum_{r\in T'}\tilde{w}(r)\le(1+\lambda)z$. The working memory bound is an immediate consequence of Theorem 2, since the working memory is dominated by the data structures from which the coreset is extracted. For what concerns the running time, observe that OUTLIERSCLUSTER can be easily implemented to run in time $O(k\cdot|T|^2)$, which is $O(k\cdot(k+z)^2)$, since $|T|=O(k+z)$ by Theorem 1. Therefore, the bound on the running time follows, since EXTRACTCORESET requires time $O\Big(\log(\log_{1+\beta}(d_{\max}/d_{\min}))\cdot(k+z)^2\Big)$ (by Theorem 2), and the while loop performs, at most, $O\Big(\log_{1+\beta}(d_{\max}/d_{\min})\Big)$ executions of OUTLIERSCLUSTER, which can be lowered to $O\Big(\log\log_{1+\beta}(d_{\max}/d_{\min})\Big)$ using binary search. □

The above result shows that allowing for a slight excess in the number of outliers governed by parameter λ results in improved space and time complexities. Note that if the upper bound z on the number of outliers must be rigidly enforced, it is sufficient to set $\lambda=1/(2z)$. In this case, $(1+\lambda)z=z+1/2$, and since the number of outliers must be an integer, it cannot be larger than z. The following corollary is an immediate consequence of Theorem 3, of this observation, and of the fact that $\log_{1+\lambda}x=\Theta((1/\lambda)\log x)$ for $\lambda\in(0,1)$.

Corollary 1. *Let $\lambda=1/(2z)$. At any time t, COMPUTESOLUTION returns a set $X\subseteq W$ of, at most, k centers, such that at least $|W|-z$ points of W are at a distance of, at most, $(23+55\beta)r^*_{k,z}(W)$ from X. The procedure requires a working memory of size $O\Big(z(k+z)\log_{1+\beta}(d_{\max}/d_{\min})\log(|W|)\Big)$ and runs in time*

$$O\Big(\log_{1+\beta}(d_{\max}/d_{\min})\cdot k(k+z)^2\Big).$$

If the while loop is substituted by a binary search for ρ_{min}, the running time decreases to

$$O\Big(\log(\log_{1+\beta}(d_{\max}/d_{\min}))\cdot k(k+z)^2\Big).$$

3.3. Obliviousness to $d_{\min}$ and $d_{\max}$

The algorithm described in Sections 3.1 and 3.2 requires the knowledge of the values $d_{\min}$ and $d_{\max}$. In this subsection, we show how to remove this requirement by employing the techniques developed in [22], suitably extended to cope with histograms, which were not used in that work.

Let $p_1,p_2,\ldots$ be an enumeration of all points of the stream S based on their arrival times. For $t>k+z$, let d_t be the minimum pairwise distance between the last $k+z+1$ points of the stream $(p_{t-k-z},\ldots,p_{t-1},p_t)$. Let also D_t be the maximum distance between p_1 and any p_i with $i\le t$, and note that, by the triangle inequality, the maximum pairwise distance among the first t points of S is upper bounded by $2D_t$. It is easy to argue that $d_t/2 \le r^*_{k+z}(W) \le r^*_{k,z}(W) \le 2D_t$. By storing p_1 and the last $k+z+1$ points of the stream, the values d_t and D_t can be straightforwardly maintained by the algorithm with $O\big((k+z)^2\big)$ operations per step. We define

$$\Gamma_t = \{(1+\beta)^i : \lfloor \log_{1+\beta} d_t/2 \rfloor \le i \le \lceil \log_{1+\beta} 2D_t \rceil\}.$$

Suppose that, at any time t, the algorithm maintains the sets A_γ, R_γ and O_γ, and the histograms for the points in $R_\gamma \cup O_\gamma$, only for $\gamma \in \Gamma_t$, and assume that the properties stated in Lemmas 1 and 2 hold for every $\gamma \in \Gamma_t$ and $r \in R_\gamma \cup O_\gamma$. Then, by running procedure EXTRACTCORESET and limiting the search for $\hat{\gamma}$ to the set Γ_t, we still obtain a $4(1+\beta)$-coreset for the current window. To see this, we first note that, based on the previous observation, Γ_t definitely includes a value γ with $r^*_{k+z}(W) \le \gamma \le (1+\beta)r^*_{k+z}(W)$. By repeating the same argument used in the proof of Theorem 1, we can show that, for such a value of γ, we have that $|A_\gamma| \le k+z$, and that the inner for loop of EXTRACTCORESET computes a set C of, at most, $k+z$ points. This immediately implies that EXTRACTCORESET determines a guess $\hat{\gamma} \le (1+\beta)r^*_{k+z}(W)$ and that the returned coreset $T = R_{\hat{\gamma}} \cup O_{\hat{\gamma}}$ is a $4(1+\beta)$-coreset.

We now show how to modify the algorithm described in the previous subsections (referred to as *full algorithm* in what follows) to maintain, without the knowledge of $d_{\min}$ and $d_{\max}$, the sets A_γ, R_γ, and O_γ and the required histograms, for every guess $\gamma \in \Gamma_t$. Suppose that this is the case up to some time $t-1 > k+z$, and consider the arrival of p_t. Before invoking UPDATE(p_t, t), the algorithm executes the operations described below.

First, the new values d_t and D_t are computed, and all sets relative to values of $\gamma \in \Gamma_{t-1} - \Gamma_t$ are removed. If $d_t < d_{t-1}$, then, for each $\gamma \in \Gamma_t$ with $\gamma < \min\{\gamma' \in \Gamma_{t-1}\} \le d_{t-1}/2$, the algorithm sets $A_\gamma = \{p_{t-k-z-1}, \ldots, p_{t-1}\} = R_\gamma$ and $O_\gamma = \emptyset$. Moreover, for each $p_\tau \in R_\gamma$, it sets $L_{p_\tau} = \{(\tau, 1)\}$, as each point represents itself only. Since any two points in A_γ are at a distance of at least $d_{t-1} > 2\gamma$, it is easy to see that these newly created data structures coincide with the ones that the full algorithm would store at time $t-1$ (for the same γ's) if the stream started at time $t-(k+z+1)$; hence, they satisfy the properties of Lemmas 1 and 2.

If $D_t > D_{t-1}$, then for each $\gamma \in \Gamma_t$ with $\gamma > \max\{\gamma' \in \Gamma_{t-1}\}$, the algorithm sets $A_\gamma = \{p_{t-|W|}\}$, $R_\gamma = \{p_{t-1}\}$ and $O_\gamma = \emptyset$. It is easy to see that these newly created sets coincide with the ones that the full algorithm would store at time $t-1$ (for the same γ's) if the stream started at time $t-|W|$; hence, they satisfy the properties of Lemma 1. It has to be remarked that, although point $p_{t-|W|}$ is not available at time $t-1$, this is not a problem since the point immediately expires at time t and is removed from the data structures without even being used in the processing of p_t. Hence, in this context, $p_{t-|W|}$ acts as a mere placeholder. For what concerns the histogram $L_{p_{t-1}}$ to associate with p_{t-1} (for every $\gamma > \max\{\gamma' \in \Gamma_{t-1}\}$), its exact version represents the entire active window, and hence it would be the list $\{(t-|W|, |W|), \ldots, (t-2, 2), (t-1, 1)\}$. In order to satisfy the properties of Lemma 2, $L_{p_{t-1}}$ can be trimmed as follows. Let $f(x) = \left\lceil \frac{x}{1+\lambda} \right\rceil$. Then, $L_{p_{t-1}}$ contains the set of pairs $(t-c_i, c_i)$, with $i \ge 0$, where the c_is form a decreasing sequence of values ≥ 1 such that $c_0 = |W|$ and, for each $i \le 0$ with $c_i > 1$, $c_{i+1} = \min\{c_i - 1, f(c_i)\}$.

Lemma 4. *The list $L_{p_{t-1}}$ defined above satisfies the properties stated in Lemma 2.*

Proof. Property 1 clearly holds. As for Property 2, consider two consecutive pairs $(t-c_i, c_i)$ and $(t-c_{i+1}, c_{i+1})$. If $c_{i+1} = f(c_i)$, then it is certain that $c_i \le (1+d)c_{i+1}$. Hence, when $c_i > (1+\lambda)c_{i+1}$, we must have $c_{i+1} = c_i - 1$, and the property follows. Property 3 holds, since, for $1 \le i \le |L_r| - 2$, $c_{i+2} \le c_{i+1} - 1 < (c_i/(1+\lambda)+1) - 1$, and hence $(1+\lambda)c_{i+2} \le c_i$. Finally, the bound stated by Property 4 follows as a direct consequence of the first three properties. □

Once the correct configurations of the data structures for all guesses $\gamma \in \Gamma_t$ are obtained, Procedure UPDATE(p_t, t) is invoked to complete step t, thus enforcing the properties of Lemmas 1 and 2 for all of these data structures.

3.4. Improved Approximation under Bounded Doubling Dimension

Consider a stream S of doubling dimension D. We now outline an improved coreset construction that is able to provide a δ-coreset T for the k-center problem with z outliers, for *any* given $\delta > 0$, at the expense of a blow-up in the working memory size, which is analyzed as a function of D and is tolerable for small (e.g., constant) D. This improved construction allows us to obtain a much tighter approximation for the k-center problem with z outliers in the sliding window setting.

Fix any given $\delta > 0$. In [22], a refinement of the k-center strategy of [21] is presented that, for every guess γ, maintains two families of attraction, representative, and orphan points. The first family, referred to as *validation points*, features three $O(k)$-sized sets of attraction, representative, and orphan points, equivalent to those described in Subection 3.1. Validation points are employed to identify a constant approximation $\hat{\gamma}$ to the optimal radius $r^*_k(W)$. The second family, referred to as *coreset points*, contains, for any guess γ, three "expanded" sets of attraction, representative, and orphan points, which refine the coverage provided by the corresponding sets of validation points, in the sense that the coreset points relative to the guess $\hat{\gamma}$ yield a coreset T such that $\max_{p \in W} d(p, T) \leq \delta r^*_k(W)$. For each γ, these larger sets contain $O(k(c/\delta)^D)$ points for a suitable constant c.

We can augment the algorithm of [22] in the same fashion as we augmented the algorithm of [21] by endowing the representative and orphan coreset points with the histograms described in Section 3.1. Then, by running the resulting algorithm for $k + z$ (instead of k) centers, the result stated in the following lemma is immediately obtained, where parameter λ and functions $w(\cdot)$ and $\tilde{w}(\cdot)$ have the same meanings as before.

Lemma 5. *Let $\delta, \lambda > 0$ be two design parameters. For a stream S of doubling dimension D, suitable data structures can be maintained, from which, at any time t, a weighted δ-coreset T of size $O((c/\delta)^D(k + z))$ for a fixed constant c can be extracted such that, for each $p \in W$, there exists a proxy $\pi(p) \in T$ with*

$$\text{dist}(p, \pi(p)) \leq \delta r^*_{k+z}(W) \leq \delta r^*_{k,z}(W).$$

Moreover, for each $r \in T$, an approximate weight $\tilde{w}(r)$ can be computed, with $w(r)/(1+\lambda) \leq \tilde{w}(r) \leq w(r)$, where $w(r) = |\{p \in W : \pi(p) = r\}|$. The data structures require a working memory of size $O\big((c/\delta)^D(k + z)\log(d_{\max}/d_{\min})\log_{1+\lambda}(|W|)\big)$.

The analysis in [22] implies that the constant c can be fixed arbitrarily close to 32. We remark that the construction described in Section 3.1 cannot return δ-coresets with $\delta \leq 4$, whereas the result of Lemma 5 yields δ-coresets for any $\delta > 0$.

To obtain the desired solution at any time t, we first compute a δ-coreset T satisfying the properties stated in the above lemma. Then, analogously to COMPUTESOLUTION, we run OUTLIERSCLUSTER(T, k, ρ, δ) for a geometric sequence of values of ρ of step $1 + \beta$ between $d_{\min}$ and $d_{\max}$, stopping at the minimum value ρ_{min} for which, if (X, T') is the output of OUTLIERSCLUSTER$(T, k, \rho_{min}, \delta)$, then the aggregate approximate weight of set T' is at most z. The algorithm returns X as the final set of (at most) k centers. By choosing $\beta = \delta/(3 + 4\delta)$, we obtain the the following result:

Theorem 4. *Let $\delta, \lambda > 0$ be two design parameters and consider a stream of doubling dimension D. There exists a sliding window algorithm that, at any time t, returns a set of, at most, k centers $X \subseteq W$, such that at least $|W| - (1 + \lambda)z$ points of W are at a distance of, at most, $(3 + 6\delta)r^*_{k,z}(W)$ from X. For a suitable constant $c > 0$, the algorithm makes use of a working memory of size $O((c/\delta)^D(k + z)\log(d_{\max}/d_{\min})\log_{1+\lambda}(|W|))$. In addition, the algorithm requires time*

$$O\Big(\log_{1+\beta}(d_{\max}/d_{\min}) \cdot \Big((c/\delta)^D(k + z) + \log_{1+\lambda}(|W|)\Big)\Big)$$

to update the data structures after each point arrival, and time

$$O\Big(\log\Big(\log_{1+\beta}(d_{\max}/d_{\min})\Big)\cdot((c/\delta)^D\cdot k(k+z))^2\Big)$$

to compute the final solution, with $\beta=\delta/(3+4\delta)$.

Proof. By reasoning as in the proof of Lemma 3, we can show that, by executing OUTLIERSCLUSTER(T,k,ρ,δ), with any $\rho \ge r^*_{k,z}(W)$, the set of uncovered points at the end of the execution has an aggregate approximate weight of, at most, z. This immediately implies that $\rho_{min} \le (1+\beta)r^*_{k,z}(W)$. Let (X,T') be the output of OUTLIERSCLUSTER(T,k,ρ_{min},δ), and let W' be the set of points of W whose proxies are in T'. We have that, for each $p\in W-W'$, $\text{dist}(\pi(p),X)\le(3+4\delta)\rho_{min}\le(3+4\delta)(1+\beta)r^*_{k,z}(W)$. Since T is an δ-coreset, we conclude that, for every $p\in W-W'$,

$$\begin{aligned}\text{dist}(p,X) &\le \text{dist}(p,\pi(p))+\text{dist}(\pi(p),X)\\ &\le \delta r^*_{k,z}(W)+(3+4\delta)(1+\beta)r^*_{k,z}(W)\\ &\le (3+6\delta)r^*_{k,z}(W),\end{aligned}$$

where the last inequality uses the fact that $\beta=\delta/(3+4\delta)$. Moreover, we also have that $|W'|\le(1+\lambda)\sum_{r\in T'}\tilde{w}(r)\le(1+\lambda)z$. Finally, the bound on the working memory follows from Lemma 5, whereas the time bounds for updating the data structures and extracting the solution from the coreset are obtained by adapting the arguments used to prove Theorems 2 and 3. □

The following corollary is the counterpart of Corollary 1 for the dimension-sensitive algorithm developed in this section.

Corollary 2. *Let $\lambda=1/(2z)$. At any time t, the algorithm is able to compute a set $X\subseteq W$ of, at most, k centers, such that at least $|W|-z$ points of W are at a distance of, at most, $(3+6\delta)r^*_{k,z}(W)$ from X. For a fixed constant $c>0$, the algorithm requires a working memory of size $O\big((c/\delta)^D z(k+z)\log(d_{\max}/d_{\min})\log(|W|)\big)$ and runs in time*

$$O\Big(\log(\log_{1+\beta}(d_{\max}/d_{\min}))(c/\delta)^D\cdot k(k+z)^2\Big),$$

with $\beta=\delta/(3+4\delta)$.

It is important to remark that the algorithm in [22] that we have built upon is *fully oblivious* to D, $d_{\max}$, and $d_{\min}$. Its augmentation discussed above inherits the obliviousness to D straightforwardly, whereas the obliviousness to $d_{\max}$ and $d_{\min}$ is inherited through the technique described in Section 3.3.

4. Effective Diameter Estimation

Consider a stream S of doubling dimension D. Building on the improved coreset construction of Lemma 5, we now outline an algorithm that, at any time t, is able to compute lower and upper estimates of the α-effective diameter Δ^α_W of the current window W. The algorithm requires the knowledge of a (possibly crude) lower bound $\eta\in(0,1)$ on the ratio between Δ^α_W and the diameter Δ_W (i.e., $\Delta^\alpha_W\ge\eta\Delta_W$).

For a given $\varepsilon>0$, let T be a weighted δ-coreset for W computed with the properties stated in Lemma 5, with $\delta=\varepsilon\eta/2$, $k=1$ and $z=0$. Hence, $|T|=O((c'/(\varepsilon\eta))^D)$, with $c'=2c$, where c is the same constant appearing the statement of the lemma. Assume for now that, for each $r\in T$, the true weight $w(r)=|\{p\in W:\pi(p)=r\}|$ is known. (Later, we will discuss the distortion introduced by using the approximate weights $\tilde{w}(r)$.) An approximation to Δ^α_W can be computed on the coreset through the following quantity:

$$\Delta^{\alpha}_{T,W} = \operatorname*{argmin}_{d}\left\{ \sum_{\substack{r_1,r_2\in T:\\ \mathrm{dist}(r_1,r_2)\le d}} w(r_1)w(r_2) \ge \alpha|W|^2 \right\}.$$

Lemma 6. $(1-\varepsilon)\Delta^{\alpha}_{W} \le \Delta^{\alpha}_{T,W} \le (1+\varepsilon)\Delta^{\alpha}_{W}$.

Proof. Let $\hat{d} = \max\{\mathrm{dist}(p,\pi(p)) : p \in W\}$. Recall that the δ-coreset T was computed with $k=1$ and $z=0$; hence, using the properties of T and the fact that $\Delta^{\alpha}_{W} \ge \eta\Delta_W$, we have that

$$\hat{d} \le \delta r^*_1(W) \le (\varepsilon\eta/2)\Delta_W \le (\varepsilon/2)\Delta^{\alpha}_{W}.$$

Using the triangle inequality, for any d and any pair $(p,q) \in W\times W$, we have that, if $\mathrm{dist}(\pi(p),\pi(q)) \le d-2\hat{d}$, then $\mathrm{dist}(p,q) \le d$. Thus, when $|\{(p,q) \in W\times W : \mathrm{dist}(\pi(p),\pi(q)) \le d-2\hat{d}\}| \ge \alpha|W|^2$, we must also have $|\{(p,q)\in W\times W : \mathrm{dist}(p,q) \le d\}| \ge \alpha|W|^2$. Consequently,

$$\begin{aligned}
\Delta^{\alpha}_{W} &= \operatorname*{argmin}_{d}\Big\{|\{(p,q)\in W\times W : \mathrm{dist}(p,q)\le d\}| \ge \alpha|W|^2\Big\}\\
&\le \operatorname*{argmin}_{d}\Big\{|\{(p,q)\in W\times W : \mathrm{dist}(\pi(p),\pi(q))\le d-2\hat{d}\}| \ge \alpha|W|^2\Big\}\\
&= \operatorname*{argmin}_{d}\left\{ \sum_{\substack{r_1,r_2\in T:\\ \mathrm{dist}(r_1,r_2)\le d-2\hat{d}}} w(r_1)w(r_2) \ge \alpha|W|^2 \right\}\\
&= \operatorname*{argmin}_{d}\left\{ \sum_{\substack{r_1,r_2\in T:\\ \mathrm{dist}(r_1,r_2)\le d}} w(r_1)w(r_2) \ge \alpha|W|^2 \right\} + 2\hat{d}\\
&= \Delta^{\alpha}_{T,W} + 2\hat{d} \le \Delta^{\alpha}_{T,W} + \varepsilon\Delta^{\alpha}_{W},
\end{aligned}$$

thus proving the first stated inequality. The proof of the other inequality is accomplished with a symmetrical argument. The triangle inequality ensures that, for every pair $(p,q) \in W\times W$, if $\mathrm{dist}(p,q) \le d$, then $\mathrm{dist}(\pi(p),\pi(q)) \le d+2\hat{d}$. Thus, when $|\{(p,q)\in W\times W : \mathrm{dist}(p,q) \le d\}| \ge \alpha|W|^2$, we must also have $|\{(p,q) \in W\times W : \mathrm{dist}(\pi(p),\pi(q)) \le d+2\hat{d}\}| \ge \alpha|W|^2$. Consequently,

$$\begin{aligned}
\Delta^{\alpha}_{W} &= \operatorname*{argmin}_{d}\Big\{|\{(p,q)\in W\times W : \mathrm{dist}(p,q)\le d\}| \ge \alpha|W|^2\Big\}\\
&\ge \operatorname*{argmin}_{d}\Big\{|\{(p,q)\in W\times W : \mathrm{dist}(\pi(p),\pi(q))\le d+2\hat{d}\}| \ge \alpha|W|^2\Big\}\\
&= \operatorname*{argmin}_{d}\left\{ \sum_{\substack{r_1,r_2\in T:\\ \mathrm{dist}(r_1,r_2)\le d+2\hat{d}}} w(r_1)w(r_2) \ge \alpha|W|^2 \right\}\\
&= \operatorname*{argmin}_{d}\left\{ \sum_{\substack{r_1,r_2\in T:\\ \mathrm{dist}(r_1,r_2)\le d}} w(r_1)w(r_2) \ge \alpha|W|^2 \right\} - 2\hat{d}\\
&= \Delta^{\alpha}_{T,W} - 2\hat{d} \ge \Delta^{\alpha}_{T,W} - \varepsilon\Delta^{\alpha}_{W}.
\end{aligned}$$

□

Recall now that, for every coreset point $r \in T$, only an approximation $\tilde{w}(r)$ to the actual weight $w(r)$ is available, with $w(r)/(1+\lambda) \leq \tilde{w}(r) \leq w(r)$. We define the approximate counterpart of $\Delta^{\alpha}_{T,W}$ as

$$\tilde{\Delta}^{\alpha}_{T,W} = \operatorname*{argmin}_{d} \sum_{\substack{r_1,r_2 \in T: \\ \mathrm{dist}(r_1,r_2) \leq d}} \tilde{w}(r_1)\tilde{w}(r_2) \geq \alpha|W|^2.$$

Our approximation algorithm returns $(1/(1+\varepsilon))\tilde{\Delta}^{\alpha/(1+\lambda)^2}_{T,W}$ and $(1/(1-\varepsilon))\tilde{\Delta}^{\alpha}_{T,W}$ as the lower and upper estimates, respectively, of the true effective diameter Δ^{α}_{W}. The following theorem establishes the tightness of these estimates and the space and time performance of the algorithm.

Theorem 5. *Consider a stream S of doubling dimension D, and a value $\alpha \in (0,1)$. Suppose that a value $\eta < 1$ is known such that, for every window W, $\Delta^{\alpha}_{W} \geq \eta\Delta_{W}$. For any $\varepsilon, \lambda > 0$, there exists a sliding window algorithm that, at any time t, is able to compute a weighted coreset T of size $O((c'/(\varepsilon\eta))^D)$, such that*

$$\frac{1}{1+\varepsilon}\tilde{\Delta}^{\alpha/(1+\lambda)^2}_{T,W} \leq \Delta^{\alpha}_{W} \leq \frac{1}{1-\varepsilon}\tilde{\Delta}^{\alpha}_{T,W},$$

where W is the current window and $c' > 0$ is a suitable constant. The algorithm makes use of a working memory of size $O\big((c'/(\varepsilon\eta))^D \log(d_{\max}/d_{\min}) \log_{1+\lambda}(|W|)\big)$. In addition, the algorithm requires time

$$O\Big(\log_{1+\beta}(d_{\max}/d_{\min}) \cdot \big((c'/(\varepsilon\eta)) + \log_{1+\lambda}(|W|)\big)\Big)$$

to update the data structures after each point arrival, where $\beta = \delta/(3+4\delta)$. The lower and upper estimates to Δ^{α}_{W} can be computed from T in time $O\big(|T|^2\big) = O\big((c'/(\varepsilon\eta))^{2D}\big)$.

Proof. We first prove that

$$\tilde{\Delta}^{\alpha/(1+\lambda)^2}_{T,W} \leq \Delta^{\alpha}_{T,W} \leq \tilde{\Delta}^{\alpha}_{T,W}.$$

Then, the stated approximation interval will immediately follow by Lemma 6. Let us first prove the leftmost inequality. From Lemma 5 we have that, for every $r \in T$, $\tilde{w}(r) \geq w(r)/(1+\lambda)$. Hence, for any d such that

$$\sum_{\substack{r_1,r_2 \in T: \\ \mathrm{dist}(r_1,r_2) \leq d}} w(r_1)w(r_2) \geq \alpha|W|^2,$$

we have that

$$\sum_{\substack{r_1,r_2 \in T: \\ \mathrm{dist}(r_1,r_2) \leq d}} \tilde{w}(r_1)\tilde{w}(r_2) \geq \frac{\alpha}{(1+\lambda)^2}|W|^2,$$

which implies $\tilde{\Delta}^{\alpha/(1+\lambda)^2}_{T,W} \leq \Delta^{\alpha}_{T,W}$. The righmost inequality is proved in a symmetrical fashion. Again, from Lemma 5, we have that, for every $r \in T$, $w(r) \geq \tilde{w}(r)$. Hence, for any d such that

$$\sum_{\substack{r_1,r_2 \in T: \\ \mathrm{dist}(r_1,r_2) \leq d}} \tilde{w}(r_1)\tilde{w}(r_2) \geq \alpha|W|^2,$$

we have that

$$\sum_{\substack{r_1,r_2 \in T: \\ \mathrm{dist}(r_1,r_2) \leq d}} w(r_1)w(r_2) \geq \alpha|W|^2,$$

which implies $\Delta^{\alpha}_{T,W} \leq \tilde{\Delta}^{\alpha}_{T,W}$. The bounds on the working memory and on the update time follow directly from Theorem 4 and from the choice of $k = 1$ and $z = 0$. Finally, the estimates

$\tilde{\Delta}_{T,W}^{\alpha/(1+\lambda)^2}$ and $\tilde{\Delta}_{T,W}^{\alpha}$ can be computed from T in time $O(|T|^2) = O((c'/(\varepsilon\eta))^{2D})$ using a simple strategy based on binary search. □

The theorem implies that by setting ε and λ sufficiently small, we can obtain tight estimates for Δ_W^{α} for all windows for which the value of the α-effective diameter behaves smoothly in an interval to the left of α. Finally, we remark that the algorithm is fully oblivious to D, $d_{\min}$, and $d_{\max}$. Moreover, while the theoretical space bound exhibits a dependency on $(1/\eta)^D$, in the next section, we provide experimental evidence of a much lesser impact of η for datasets where outliers represent true noise, proving that the working space requirements exhibit a milder dependence on the crudeness of the lower bound η.

5. Experiments

We implemented the algorithms for k-center with z outliers and for the estimation of the effective diameter presented in Sections 3 and 4. For what concerns k-center with z outliers, we implemented the dimensionality-sensitive algorithm of Section 3.4, which offers a wider spectrum of performance–accuracy tradeoffs. We ran proof-of-concept experiments aimed at testing the algorithms' behavior against relevant competitors in terms of approximation, memory usage, and running time for processing each point arrival (*update time*) and for computing a solution for the current window whenever needed (*query time*). All tests were executed using Java 13 on a Windows machine running on an AMD FX8320 processor with 12GB of RAM, with the running times measured using `System.nanoTime`, and with the points to the algorithms being fed through the file input stream.

5.1. *k-Center with Outliers*

Along with our algorithm (dubbed OUR-SLIDING), we implemented the sequential 3-approximation by [7] (dubbed CHARIKAR) to be run on the entire window W, consisting of a search for the minimum ρ such that OUTLIERSCLUSTER$(W, k, \rho, 0)$, run with unit weights, ends with, at most, z uncovered points. We chose this sequential benchmark over the existing LP-based 2-approximation algorithms since these latter algorithms do not seem to admit practical implementations. Since CHARIKAR itself is plagued by a superquadratic complexity, which makes it unfeasible for larger windows, we also devised a sampled version (dubbed SAMP-CHARIKAR) where the center selection in each call to OUTLIERSCLUSTER examines only a fixed number of random candidates, rather than all window points. We deemed it unnecessary to perform a comparison of our algorithm with the one in [23], since, as mentioned in the introduction, this latter algorithm needs to run an instance of OUTLIERSCLUSTER for each update operation, and would thus prove to be a poor competitor of our strategy, where the execution of this expensive sequential procedure is confined only to the query operation.

Also, to assess the importance of using a specialized algorithm to handle outliers, we compared the quality of our solution against the one returned by the algorithm of [12] for k-center without outliers (dubbed GON), where the radius is computed excluding the z largest distances from the centers.

The algorithms were tested on the following datasets, often used in previous works [8]: the `Higgs` dataset (http://archive.ics.uci.edu/ml/datasets/HIGGS), (accessed on 10 January 2022), which contains 11 million seven-dimensional points representing high-energy particle features generated through Monte-Carlo simulations; and the `Cover` dataset (https://archive.ics.uci.edu/ml/datasets/covertype), (accessed on 10 January 2022), which contains 581,012 55-dimensional points from geological observations of US forest biomes, and was employed as a stress test for our dimensionality-sensitive algorithm. We also generated inflated versions of the original datasets, dubbed `Higgs+` and `Cover+`, by artificially injecting a new true outlier point after each original point with probability p, where the new point has a norm 100 times the diameter of the original dataset (e.g., as if produced by a malfunctioning sensor). The probability p was chosen to yield $z/2$ true outliers per window, in expectation. We performed tests for $k = 10$, $z = 10, 50$, and window sizes $|W| = N \in \{10^4, 10^5, 10^6\}$ using Euclidean distance.

For OUR-SLIDING, we set $\delta = 2/3$, $\beta = 0.5$, and $\lambda = 0.5$; moreover, we set $d_{\min} = 0.01$ and $d_{\max} = 10^4$, which are conservative lower and upper estimates of the clustering radii for all windows and all datasets. The implementations of CHARIKAR and SAMP-CHARIKAR execute, for a window W, a search for a minimum ρ such that OUTLIERSCLUSTER$(W, k, \rho, 0)$ with unit weights ends with $\leq z$ uncovered points. For SAMP-CHARIKAR, OUTLIERSCLUSTERS has been modified so that each new center is selected among a set of random window points of expected size 1000.

Tables 1 and 2 detail the full results of the experiments on k-center clustering with z outliers. All quantities are provided as one-sigma confidence intervals, based on 10 windows sampled every 10^4 timesteps after the first N insertions. Starred results are based on a single sample due to the excessively high running time. Table 1 reports the average ratio between the clustering radius obtained by each tested algorithm and the one obtained by OUR-SLIDING, as well as the average number of floats maintained in memory by the algorithms. (All radii have been computed with respect to the entire window, excluding the z largest distances from the centers.) As shown in the table, OUR-SLIDING is always within a few percentage points from the radius of the solution of CHARIKAR (which did not finish in reasonable time for $N = 10^6$). On the other hand, the quality of the solution of SAMP-CHARIKAR degrades as the window size grows, since the fraction of center candidates decreases. Moreover, as expected, GON yields a poorer performance, especially in the presence of true outliers, as these are mistakenly selected as centers instead of being disregarded. Hence, GON is not considered in the successive experiments. Figure 1 plots the memory usage (in floats) of the algorithms for `Higgs`, confirming that the working memory required by OUR-SLIDING grows sublinearly with N, and it is much smaller than the one required by CHARIKAR and SAMP-CHARIKAR, which is linear in N.

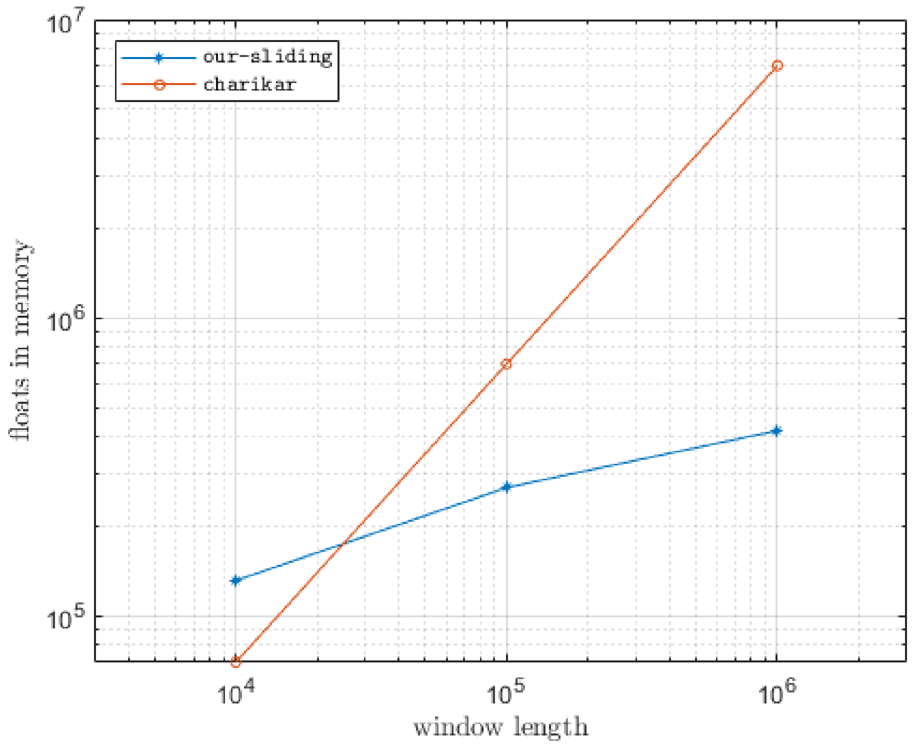

Figure 1. Working memory (`Higgs`, $z = 10$).

Table 1. Comparison between clustering radii and between working memory requirements.

Dataset	Algorithm	Obj. Ratio			Memory ($\times 10^6$ Floats)		
		Window Size			Window Size		
		10^4	10^5	10^6	10^4	10^5	10^6
HIGGS (z = 10)	OUR-SLIDING	1 ± 0	1 ± 0	1 ± 0	0.13 ± 0.01	0.27 ± 0.02	0.42 ± 0.02
	CHARIKAR	1.02 ± 0.05	0.99 ± 0.04	-	0.07 ± 0	0.7 ± 0	7 ± 0
	SAMP-CHARIKAR	1.07 ± 0.05	1.43 ± 0.23	2.74 ± 0.7	0.07 ± 0	0.7 ± 0	7 ± 0
	GON	1.18 ± 0.12	1.17 ± 0.08	1.05 ± 0.03	0.07 ± 0	0.7 ± 0	7 ± 0
HIGGS (z = 50)	OUR-SLIDING	1 ± 0	1 ± 0	1 ± 0	0.26 ± 0.01	0.65 ± 0.03	1.25 ± 0.02
	CHARIKAR	1.02 ± 0.04	-	-	0.07 ± 0	0.7 ± 0	7 ± 0
	SAMP-CHARIKAR	1.04 ± 0.06	1.14 ± 0.08	1.59 ± 0.15	0.07 ± 0	0.7 ± 0	7 ± 0
	GON	1.54 ± 0.19	1.5 ± 0.15	1.19 ± 0.06	0.07 ± 0	0.7 ± 0	7 ± 0
COVER (z = 10)	OUR-SLIDING	1 ± 0	1 ± 0		0.72 ± 0.18	2.06 ± 0.13	
	CHARIKAR	0.97 ± 0.17	1.02 *		0.55 ± 0	5.5 ± 0	
	SAMP-CHARIKAR	0.96 ± 0.17	0.98 ± 0.18		0.55 ± 0	5.5 ± 0	
	GON	1.1 ± 0.17	0.94 ± 0.15		0.55 ± 0	5.5 ± 0	
COVER (z = 50)	OUR-SLIDING	1 ± 0	1 ± 0		1.83 ± 0.22	5.38 ± 0.22	
	CHARIKAR	0.98 ± 0.09	-		0.55 ± 0	5.5 ± 0	
	SAMP-CHARIKAR	0.99 ± 0.1	1.04 ± 0.2		0.55 ± 0	5.5 ± 0	
	GON	1.13 ± 0.11	1.02 ± 0.17		0.55 ± 0	5.5 ± 0	
HIGGS+ (z = 10)	OUR-SLIDING	1 ± 0	1 ± 0	1 ± 0	0.12 ± 0.01	0.26 ± 0.02	0.43 ± 0.02
	CHARIKAR	0.98 ± 0.18	0.97 ± 0.04	-	0.07 ± 0	0.7 ± 0	7 ± 0
	SAMP-CHARIKAR	1.09 ± 0.17	1.61 ± 0.2	3.03 ± 0.51	0.07 ± 0	0.7 ± 0	7 ± 0
	GON	1.63 ± 0.29	1.3 ± 0.16	1.4 ± 0.02	0.07 ± 0	0.7 ± 0	7 ± 0
COVER+ (z = 10)	OUR-SLIDING	1 ± 0	1 ± 0		0.65 ± 0.17	1.93 ± 0.21	
	CHARIKAR	0.99 ± 0.15	1.02 *		0.55 ± 0	5.5 ± 0	
	SAMP-CHARIKAR	0.97 ± 0.14	0.96 ± 0.15		0.55 ± 0	5.5 ± 0	
	GON	3.07 ± 2.02	1.44 ± 0.22		0.55 ± 0	5.5 ± 0	

Table 2 reports update times (in milliseconds) and query times (in seconds). For OUR-SLIDING, the update time is the time required to process any newly arrived point, whereas the query time includes the time to extract the coreset and compute the final solution on the coreset. For CHARIKAR and SAMP-CHARIKAR, the update time is null, whereas the query time is the time taken to extract the solution from the whole window. The running times reveal that, by virtue of the coreset-based approach, OUR-SLIDING features a query time that is much smaller than the one of CHARIKAR and SAMP-CHARIKAR. The update times for OUR-SLIDING, although not negligible, are three orders of magnitude smaller than the query times. The experiments for $z = 50$ show that, whereas the working memory requirements and running times increase with z, for reasonably large values of N, OUR-SLIDING still exhibits a much lower memory footprint than CHARIKAR and still returns solutions of comparable quality.

Table 2. Comparison between update times and between query times.

Dataset	Algorithm	Update Time (ms)			Query Time (s)		
		Window Size			Window Size		
		10^4	10^5	10^6	10^4	10^5	10^6
HIGGS (z = 10)	OUR-SLIDING	0.66 ± 0.17	1 ± 0.44	1.98 ± 0.7	2.52 ± 1.22	5.5 ± 0.87	8.2 ± 0.42
	CHARIKAR	0 ± 0	0 ± 0	0 ± 0	24.88 ± 2.52	3080.8 ± 160.8	-
	SAMP-CHARIKAR	0 ± 0	0 ± 0	0 ± 0	2.51 ± 0.26	31.5 ± 1.34	297.8 ± 30.9
HIGGS (z = 50)	OUR-SLIDING	1.24 ± 0.66	4.03 ± 0.53	5.39 ± 1.14	10.51 ± 0.76	93.22 ± 24.76	75.99 ± 10.38
	CHARIKAR	0 ± 0	0 ± 0	0 ± 0	27.97 ± 2.68	–	–
	SAMP-CHARIKAR	0 ± 0	0 ± 0	0 ± 0	2.87 ± 0.24	47.95 ± 2.11	384.4 ± 90.92
COVER (z = 10)	OUR-SLIDING	1.05 ± 0.46	3.74 ± 0.57		12.85 ± 2.59	87.82 ± 46.08	
	CHARIKAR	0 ± 0	0 ± 0		314.93 ± 55.21	$5 \cdot 10^4$ *	
	SAMP-CHARIKAR	0 ± 0	0 ± 0		32.1 ± 5.25	400.81 ± 66.89	
COVER (z = 50)	OUR-SLIDING	3.31 ± 0.8	11.22 ± 1.28		44.06 ± 6.88	892.82 ± 654.3	
	CHARIKAR	0 ± 0	0 ± 0		310.87 ± 43.6	-	
	SAMP-CHARIKAR	0 ± 0	0 ± 0		31.06 ± 3.05	431.18 ± 28.42	
HIGGS+ (z = 10)	OUR-SLIDING	0.65 ± 0.26	1.22 ± 0.37	1.89 ± 0.42	0.92 ± 0.93	5.28 ± 1.82	8.37 ± 0.31
	CHARIKAR	0 ± 0	0 ± 0	0 ± 0	27.53 ± 3.53	3224.2 ± 252.4	–
	SAMP-CHARIKAR	0 ± 0	0 ± 0	0 ± 0	2.7 ± 0.22	35.34 ± 3.14	328.1 ± 36.08
COVER+ (z = 10)	OUR-SLIDING	1.21 ± 0.5	3.14 ± 0.6		7.08 ± 6.54	51.22 ± 39.68	
	CHARIKAR	0 ± 0	0 ± 0		402.32 ± 71.95	$4 \cdot 10^4$ *	
	SAMP-CHARIKAR	0 ± 0	0 ± 0		41.47 ± 9.44	418.03 ± 47.58	

We also tested the sensitivity of our algorithm's performance to the λ parameter, making it range in $[0, 1]$. The results for `Higgs` are shown in Figure 2 and in Tables 3 and 4. Specifically, Table 3 reports on the sensitivity of the clustering radius and of the memory requirements, whereas Table 4 reports on the sensitivity of the update and query times. The experiments were run on `Higgs` and `Cover`, with $k = z = 10$, setting, as before, $\delta = 2/3$, $\beta = 0.5$, $d_{\min} = 0.01$, $d_{\max} = 10^4$. For λ, we used the values: $0, 0.1, 0.5, 1$. As shown in Figure 2, setting $\lambda = 0$ (i.e., maintaining the full histograms) leads to an unbearable increase in memory usage, and, hence, in execution times. With approximate histograms (i.e., $\lambda > 0$), the memory usage decreases as λ increases, with a significant drop already for $\lambda = 0.1$. While $\lambda = 0$ yields the solution with the best approximation, in our tests, we often obtained the same solution using $\lambda = 0.1$. Most importantly, the degradation of the clustering radius never exceeded 1%, even for $\lambda = 1$.

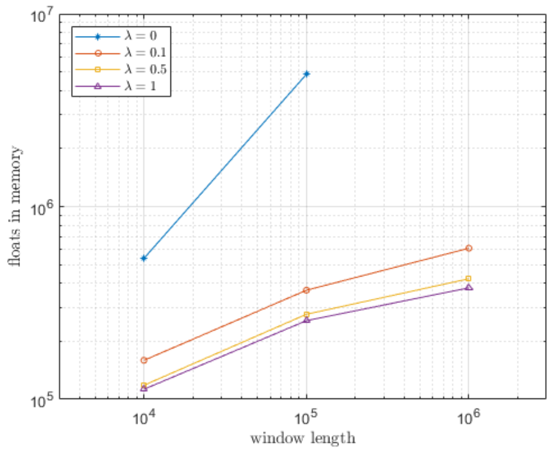

Figure 2. OUR-SLIDING: sensitivity to λ (`Higgs`, $z = 10$).

Table 3. Sensitivity of OUR-SLIDING to λ: clustering radii and working memory requirements.

Dataset	**Algorithm**	**Clustering Radius**			**Memory ($\times 10^6$ Floats)**		
		Window Size			**Window Size**		
		$\mathbf{10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$	$\mathbf{10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
HIGGS (z = 10)	$\lambda = 0$	2.826 ± 0.178	4.289 ± 0.162	-	0.53 ± 0.04	4.85 ± 0.96	-
	$\lambda = 0.1$	2.826 ± 0.178	4.289 ± 0.162	5.981 ± 0.124	0.15 ± 0.01	0.36 ± 0.03	0.6 ± 0.02
	$\lambda = 0.5$	2.808 ± 0.195	4.297 ± 0.165	6.031 ± 0.027	0.13 ± 0.01	0.27 ± 0.02	0.42 ± 0
	$\lambda = 1$	2.812 ± 0.178	4.281 ± 0.176	6.031 ± 0.021	0.12 ± 0.01	0.24 ± 0.02	0.37 ± 0
COVER (z = 10)	$\lambda = 0$	1177.56 ± 400.03	1976.02 ± 150.66		0.75 ± 0.17	2.3 ± 0.15	
	$\lambda = 0.1$	1177.56 ± 400.03	1973.31 ± 149.53		0.74 ± 0.17	2.23 ± 0.14	
	$\lambda = 0.5$	1177.56 ± 400.03	1954.08 ± 145.71		0.72 ± 0.18	2.06 ± 0.13	
	$\lambda = 1$	1180.44 ± 405.86	1967.02 ± 150.61		0.71 ± 0.18	2 ± 0.12	

Table 4. Sensitivity of OUR-SLIDING to λ: update and query times.

Dataset	**Algorithm**	**Update Time (ms)**			**Query Time (s)**		
		Window Size			**Window Size**		
		$\mathbf{10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$	$\mathbf{10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
HIGGS (z = 10)	$\lambda = 0$	3.49 ± 0.69	35.18 ± 21.11	-	2.49 ± 1.16	5.91 ± 0.74	-
	$\lambda = 0.1$	0.72 ± 0.38	1.56 ± 0.53	2.58 ± 0.41	2.47 ± 1.18	5.88 ± 0.89	8.88 ± 0.47
	$\lambda = 0.2$	0.57 ± 0.12	1.39 ± 0.42	2.06 ± 0.67	2.46 ± 1.19	5.82 ± 0.91	7.74 ± 0.25
	$\lambda = 1$	0.57 ± 0.1	1.32 ± 0.36	1.99 ± 0.32	2.47 ± 1.2	5.81 ± 0.92	7.77 ± 0.15
COVER (z = 10)	$\lambda = 0$	1.12 ± 0.38	3.75 ± 0.32		12.44 ± 2.38	90.65 ± 51.26	
	$\lambda = 0.1$	1.33 ± 0.69	3.72 ± 0.53		12.31 ± 1.92	90.36 ± 51.44	
	$\lambda = 0.2$	1.27 ± 0.5	3.57 ± 0.25		12.41 ± 2.19	90 ± 51.14	
	$\lambda = 1$	1.21 ± 0.5	4.29 ± 2.28		12.32 ± 2.17	90.34 ± 50.61	

Overall, the experiments confirm that OUR-SLIDING is able to achieve a precision comparable to the sequential algorithms at a fraction of their memory/time requirements.

5.2. *Effective Diameter*

We compared our algorithm for estimating the effective diameter described in Section 4 (dubbed EFF-SLIDING) against the following sequential baseline (dubbed EFF-SEQUENTIAL). EFF-SEQUENTIAL computes all N^2 distances in the window W and, to avoid storing all of them, only keeps track of how many distances lay in each interval $[d_{\min} \cdot (1+\rho)^i, d_{\min} \cdot (1+\rho)^{i+1}]$, for $i \geq 0$, by maintaining the appropriate counters. We set $\rho = 0.01$ so that the error due to this discretization is minimal. After all distances have been computed, the algorithm sweeps the counters and returns the minimum value $d_{\min} \cdot (1+\rho)^i$ for which at least $\lceil \alpha |W|^2 \rceil$ distances fall below that value. This same procedure, adapted to account for weights, is also used in EFF-SLIDING to compute the solution on the weighted coreset.

We experimented on the `Higgs-eff` dataset, which is another artificially inflated version of the `Higgs` dataset where a true outlier (i.e., a random point whose norm is 100 times the diameter of the original dataset) is injected, on average, every 1000 points. In the experiment, we set $\alpha = 0.9$. Since, for every tested window size N, $\alpha N^2 \ll (N - N/1000)^2$, we expect, in this controlled experiment, the α-effective diameter of each window of `Higgs-eff` to be close to the diameter of the non-outlier points in the window. For our algorithm, we set $\varepsilon = 5/3$ and $\lambda = \beta = 0.5$. Moreover, we set $d_{\min} = 0.01$ and $d_{\max} = 10^4$. Finally, we set $\eta = 1/1000$, which is a very conservative lower bound to the ratio between the effective diameter and the diameter of the dataset for any window W.

The results of these experiments are reported in Tables 5 and 6. Table 5 reports the ratio between the effective diameter computed by EFF-SEQUENTIAL and the (conservative) upper estimate $\tilde{\Delta}^{\alpha}_{T,W}$ computed by EFF-SLIDING, as well as the average number of floats maintained in memory by the algorithms. As shown in the table, the solution returned by EFF-SLIDING is almost indistinguishable from the one returned by EFF-SEQUENTIALfor those window sizes for which EFF-SEQUENTIAL, whose complexity grows quadratically, could be executed within reasonable times. On the other hand, the memory usage of EFF-SLIDING grows very slowly with N, and, thus, for large enough window sizes, becomes lower than the one of EFF-SEQUENTIAL. Table 6 reports the update and query times for the two algorithms. Due to the reduced coreset size, the query times of EFF-SLIDING are orders of magnitude lower than those of EFF-SEQUENTIAL, and the updated times of EFF-SLIDING, although not negligible, are significantly smaller than the query times.

Table 5. Effective diameter: comparison between estimates and between working memory requirements.

Dataset	**Algorithm**	**Diameter Ratio**			**Memory ($\times 10^6$ floats)**		
		Window Size			**Window Size**		
		10^4	**10^5**	**10^6**	**10^4**	**10^5**	**10^6**
HIGGS-EFF	EFF-SLIDING	1 ± 0	1 ± 0	1 ± 0	0.52 ± 0.36	0.41 ± 0.23	0.53 ± 0.1
	EFF-SEQUENTIAL	0.991 ± 0.019	0.992 ± 0.009	-	0.07 ± 0	0.7 ± 0	7 ± 0

Table 6. Effective diameter: comparison between update times and between query times.

Dataset	**Algorithm**	**Update Time (ms)**			**Query Time (s)**		
		Window Size			**Window Size**		
		10^4	**10^5**	**10^6**	**10^4**	**10^5**	**10^6**
HIGGS-EFF	EFF-SLIDING	3.63 ± 3.76	2.53 ± 1.57	3.33 ± 1.16	0.05 ± 0.01	0.77 ± 0.16	7.41 ± 0.57
	EFF-SEQUENTIAL	0 ± 0	0 ± 0	0 ± 0	9.26 ± 1.23	994.81 ± 38.63	-

Finally, we tested on tailor-made artificial datasets the impact of the parameter η, which is a (possibly crude) lower bound on the ratio between the effective diameter and the diameter. In fact, the theoretical bounds on the coreset size embody a factor proportional to $(1/\eta)^D$, which could lead to a severe deterioration of the performance indicators for low (i.e., conservative) values of η. In reality, for datasets where the discrepancy between the diameter and effective diameter is caused by few distant outliers (noisy points), since the balls centered on coreset points have radius $O(\varepsilon\eta\Delta_W) = O(\varepsilon\Delta^{\alpha}_{W})$ and since most of the points will be contained in a ball of radius $O(\Delta^{\alpha}_{W})$, with only a few outliers at distance $O(\Delta_W)$, the actual number of points maintained in the coreset should not really depend on η. To test this intuition, we created artificial datasets by generating random points in a ball of unit radius with a few outliers (one every 1000 points, on average) on the surface of a ball of radius R for values of R in $\{10, 100, 1000\}$. We set $\alpha = 0.9$ as before, and ran our algorithm with $\varepsilon = 10/3$ and $\lambda = 0.5$. Moreover, we set $\eta = 1/(2R)$ as it lower bounds the ratio between the effective diameter and the diameter. We report the results for window size $N = 10^5$, since a similar pattern emerges for other values of N. Indeed, as Table 7 shows, the memory usage is, in practice, almost constant across all values of η.

Table 7. Effective diameter: sensitivity to η.

Dataset	Eff. Diameter	Memory ($\times 10^6$ floats)	Update Time (ms)	Query Time (s)
$R = 10$	1.175 ± 0	0.22 ± 0.05	0.91 ± 0.31	1.69 ± 0.5
$R = 100$	1.175 ± 0.006	0.24 ± 0.11	1.79 ± 1.01	0.84 ± 0.25
$R = 1000$	1.178 ± 0.006	0.35 ± 0.14	1.9 ± 1.39	4.2 ± 1.34

6. Conclusions

In this paper, we have presented coreset-based streaming algorithms for the k-center problem with z outliers and for the estimation of the α-effective diameter under the sliding window setting. Our algorithms require a working memory that is considerably smaller than the window size, and, with respect to the state-of-the-art sequential algorithms executed on the entire window, they are up to orders of magnitude faster while achieving a comparable accuracy. The effectiveness of our approach has been confirmed by a set of proof-of-concept experiments on both real-world and synthetic datasets.

Based on the theoretical analysis conducted in the paper, the space and time required by our algorithms to attain a high accuracy seem to grow steeply (in fact, exponentially) with the doubling dimension of the stream. An interesting, yet challenging, research avenue is to investigate whether this steep dependence can be ameliorated by means of alternative techniques (e.g., the use of randomization). In addition, it would be interesting to incorporate some notion of local density in the clustering quality measure; for example, by using a robust distance measure, such as the one proposed in [29].

Author Contributions: Conceptualization: P.P., A.P. and G.P.; formal analysis and investigation: P.P., A.P. and G.P.; writing—original draft preparation: P.P.; writing—review and editing: A.P. and G.P. All authors have read and agreed to the published version of the manuscript.

Funding: 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".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code and the datasets used in our experiments are provided on GitHub at github.com/PaoloPellizzoni/OutliersSlidingWindows, (accessed on 10 January 2022).

Acknowledgments: The authors wish to thank the anonymous referees for their constructive criticism that helped to improve the quality of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Henzinger, M.; Raghavan, P.; Rajagopalan, S. Computing on Data Streams. In Proceedings of the DIMACS Workshop on External Memory Algorithms, New Brunswick, NJ, USA, 20–22 May 1998; pp. 107–118.
2. Datar, M.; Motwani, R. The Sliding-Window Computation Model and Results. In *Data Stream Management*; Garofalakis, M., Gehrke, J., Rastogi, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 149–165.
3. Braverman, V. Sliding Window Algorithms. In *Encyclopedia of Algorithms*; Cao, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 2006–2011.
4. Snyder, L. Introduction to facility location. In *Wiley Enciclopedia of Operations Research and Management Science*; Wiley: Hoboken, NJ, USA, 2011.
5. Hennig, C.; Meila, M.; Murtagh, F.; Rocci, R. *Handbook of Cluster Analysis*; CRC Press: Boca Raton, FL, USA, 2015.
6. Bateni, M.; Esfandiari, H.; Fischer, M.; Mirrokni, V. Extreme k-Center Clustering. In Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI), Vancouver, BC, Canada, 2–9 February 2021; pp. 3941–3949.
7. Charikar, M.; Khuller, S.; Mount, D.; Narasimhan, G. Algorithms for Facility Location Problems with Outliers. In Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), Washington, DC, USA, 7–9 January 2001; pp. 642–651.

8. Malkomes, G.; Kusner, M.; Chen, W.; Weinberger, K.; Moseley, B. Fast Distributed k-Center Clustering with Outliers on Massive Data. In Proceedings of the 28th Conference on Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 7–12 December 2015; pp. 1063–1071.
9. Ceccarello, M.; Pietracaprina, A.; Pucci, G. Solving k-center Clustering (with Outliers) in MapReduce and Streaming, almost as Accurately as Sequentially. *arXiv* **2018**, arXiv:1802.09205.
10. Rousseeuw P.J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
11. Altieri, F.; Pietracaprina, A.; Pucci, G.; Vandin, F. Scalable Distributed Approximation of Internal Measures for Clustering Evaluation. In Proceedings of the SIAM International Conference on Data Mining (SDM), Online, 29 April–1 May 2021; pp. 648–656.
12. Gonzalez, T.F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.* **1985**, *38*, 293–306. [CrossRef]
13. Hochbaum, D.; Shmoys, D. A Best Possible Heuristic for the *k*-Center Problem. *Math. Oper. Res.* **1985**, *10*, 180–184. [CrossRef]
14. Chan, T.H.H.; Guerqin, A.; Sozio, M. Fully Dynamic k-Center Clustering. In Proceedings of the World Wide Web Conference, Lyon, France, 23–27 April 2018; Volume 2018, pp. 579–587.
15. Harris, D.; Pensyl, T.; Srinivasan, A.; Trinh, K. A Lottery Model for Center-Type Problems With Outliers. *ACM Trans. Algorithms* **2019**, *15*, 36. [CrossRef]
16. Chakrabarty, D.; Goyal, P.; Krishnaswamy, R. The Non-Uniform *k*-Center Problem. *ACM Trans. Algorithms* **2020**, *16*, 46. [CrossRef]
17. Ding, H.; Yu, H.; Wang, Z. Greedy Strategy Works for k-Center Clustering with Outliers and Coreset Construction. In Proceedings of the 27th Annual European Symposium on Algorithms (ESA), Munich/Garching, Germany, 9–11 September 2019; pp. 40:1–40:16.
18. McCutchen, R.; Khuller, S. Streaming Algorithms for k-Center Clustering with Outliers and with Anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 165–178.
19. Feldmann, A.; Marx, D. The Parameterized Hardness of the k-Center Problem in Transportation Networks. *Algorithmica* **2020**, *82*, 1989–2005 [CrossRef]
20. Cohen-Addad, V.; Feldmann, A.; Saulpic, D. Near-Linear Time Approximation Schemes for Clustering in Doubling Metrics. *J. ACM* **2021**, *68*, 1–34 [CrossRef]
21. Cohen-Addad, V.; Schwiegelshohn, C.; Sohler, C. Diameter and k-Center in Sliding Windows. In Proceedings of the 43th International Colloquium on Automata, Languages and Programming (ICALP), Rome, Italy, 11–15 July 2016; pp. 19:1–19:12.
22. Pellizzoni, P.; Pietracaprina, A.; Pucci, G. Dimensionality-adaptive k-center in sliding windows. In Proceedings of the 7th IEEE International Conference on Data Science and Advanced Analytics (DSAA), Sydney, Australia, 6–9 October 2020; pp. 197–206.
23. de Berg, M.; Monemizadeh, M.; Zhong, Y. k-Center Clustering with Outliers in the Sliding-Window Model. In Proceedings of the 29th Annual European Symposium on Algorithms (ESA), Lisbon, Portugal, 29–30 September 2021; pp. 13:1–13:13.
24. Braverman, V.; Lang, H.; Levin, K.; Monemizadeh, M. Clustering Problems on Sliding Windows. In Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Arlington, VA, USA, 10–12 January 2016; pp. 1374–1390.
25. Borassi, M.; Epasto, A.; Lattanzi, S.; Vassilvitskii, S.; Zadimoghaddam, M. Sliding Window Algorithms for k-Clustering Problems. In Proceedings of the 34th Neural Information Processing Systems (NeurIPS), Online, 6–12 December 2020.
26. Palmer, C.; Gibbons, P.; Faloutsos, C. ANF: A fast and scalable tool for data mining in massive graphs. In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Edmonton, AB, Canada, 23–26 July 2002; pp. 81–90.
27. Braverman, V.; Ostrovsky, R. Smooth Histograms for Sliding Windows. In Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Providence, RI, USA, 20–23 October 2007; pp. 283–293.
28. Gottlieb, L.A.; Kontorovich, A.; Krauthgamer, R. Efficient Classification for Metric Data. *IEEE Trans. Inf. Theory* **2014**, *60*, 5750–5759. [CrossRef]
29. Hu, L.; Zhong, C. An Internal Validity Index Based on Density-Involved Distance. *IEEE Access* **2019**, *7*, 40038–40051. [CrossRef]