Geometric Optimization Problems over Sliding Windows*

Timothy M. Chan and Bashir S. Sadjad

School of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{tmchan,bssadjad}@uwaterloo.ca

Abstract. We study the problem of maintaining a $(1+\epsilon)$-factor approximation of the *diameter* of a stream of points under the *sliding window* model. In one dimension, we give a simple algorithm that only needs to store $O(\frac{1}{\epsilon}\log R)$ points at any time, where the parameter R denotes the "spread" of the point set. This bound is optimal and improves Feigenbaum, Kannan, and Zhang's recent solution by two logarithmic factors. We then extend our one-dimensional algorithm to higher constant dimensions and, at the same time, correct an error in the previous solution. In high nonconstant dimensions, we also observe a constant-factor approximation algorithm that requires sublinear space. Related optimization problems, such as the *width*, are also considered in the two-dimensional case.

1 Introduction

In conventional settings an algorithm has access to the whole input at once. In the *data stream* model [12, 15], however, data elements come in one by one. In this model, we want to maintain some function over the input (for example, statistical information) but because of space limitations, we are not allowed to store all data in memory. In the more general *sliding window* model, the setting is similar but the goal function should be maintained over a window containing the N newest elements in the stream. In some cases the value of N is fixed in advance, but in some applications the size of the window may be dynamic. (For example, we may want to maintain the goal function over data received within the last one hour.)

The objective of this paper is to study some well-known optimization problems in computational geometry, *e.g.*, the *diameter* and the *width*, under the sliding window model. For a set P of n points in d-dimensional space, the diameter $\Delta(P)$ is the distance between the furthest pair of points in P. The width of P is the minimum distance between two parallel hyperplanes where all points lie between them. We focus primarily on the case where d is a small constant.

* Work done by the first author was supported by an NSERC Research Grant and a Premiere's Research Excellence Award. This work has appeared as part of the second author's Master's thesis.

The diameter problem has been studied extensively in the traditional model. Although $O(n \log n)$ algorithms have been given for $d = 2$ [16] and $d = 3$ [8, 17], only slightly subquadratic algorithms are known for higher dimensions. This has prompted researchers [2, 4, 6] to consider approximation algorithms. For example, with the latest algorithm [7], a $(1+\epsilon)$-factor approximation of the diameter can be found in $O(n+(\frac{1}{\epsilon})^{d-1.5})$ time. Earlier Agarwal et al. [2] observed a simple approximation algorithm that runs in $O((\frac{1}{\epsilon})^{(d-1)/2}n)$ time and is in fact a data-stream algorithm requiring just $O((\frac{1}{\epsilon})^{(d-1)/2})$ space. Other data-stream approximation algorithms in the two-dimensional case were considered by Feigenbaum et al. [10] and Hershberger and Suri [13]. (No efficient data-stream exact algorithm is possible even in one dimension [10].) For the high-dimensional case, where d is not a fixed constant, Goel et al. [11] have proposed a $(1+\epsilon)$-approximation algorithm in the traditional model that runs in $O(n^{1+1/(1+\epsilon)} + dn)$ time, and Indyk [14] has developed a $(c+\epsilon)$-approximation algorithm in the data stream model with sublinear space complexity $O(dn^{1/(c^2-1)} \log n)$ for $c > \sqrt{2}$.

The width problem has also been extensively studied in the traditional model, both in terms of exact algorithms [3, 16] and approximation algorithms [6, 9]. The recent approximation algorithm by Chan [7], for example, runs in $O(n+(\frac{1}{\epsilon})^{d-1})$ time for any fixed d. An efficient approximation algorithm in the data stream model has also been proposed by Chan, using only $O((\frac{1}{\epsilon}\log\frac{1}{\epsilon})^{d-1})$ space.

The previous (and perhaps the only) approximate diameter algorithm in the sliding window model was proposed by Feigenbaum et al. [10]. Their one-dimensional algorithm (briefly reviewed in Section 2) requires $O(\frac{1}{\epsilon}\log^3 N(\log R + \log\log N + \log\frac{1}{\epsilon}))$ bits of space, where N is the size of the sliding window and R is the *spread* of the points in the window; the spread is defined as the ratio between the diameter and the minimum non-zero distance between any two points in the window. Also Feigenbaum et al. have shown a lower bound of $\Omega(\frac{1}{\epsilon}\log R \log N)$ bits of space. They also claimed that their one-dimensional algorithm can be used to approximate the diameter in any fixed dimension d, with $O((\frac{1}{\epsilon})^{(d+1)/2}\log^3 N(\log R + \log\log N + \log\frac{1}{\epsilon}))$ bits of space.

To our knowledge, there is no existing solution for the width problem in the sliding window model.

In this paper, we give a number of new results in the sliding window model:

- For diameter in one dimension (Section 3), we present a simpler and more efficient algorithm that stores only $\Theta(\frac{1}{\epsilon}\log R)$ points. Under the assumption (made also by Feigenbaum et al.) that coordinates can be encoded with $O(\log N)$ bits, our algorithm thus uses $O(\frac{1}{\epsilon}\log R \log N)$ bits of space, matching the known lower bound. Our algorithm is also faster, requiring $O(1)$ time per point (Feigenbaum et al.'s method requires $O(\log N)$ time per point).
- For diameter in a fixed dimension (Section 4), we point out a small error in Feigenbaum et al.'s previous solution. In addition, we show that our one-dimensional algorithm can be extended to higher dimensions. The number of points stored is $O((\frac{1}{\epsilon})^{(d+1)/2}\log\frac{R}{\epsilon})$, which again improves the previous (claimed) bound.

– For diameter in a higher non-constant dimension, or more generally, in a metric space (Section 5), we mention a $(6+\epsilon)$-approximation algorithm that uses sublinear space. (Although a constant-factor result is trivial for the data stream model, it is not as obvious for sliding windows.)
– We also give the first solution for the width problem in the sliding window model in the two-dimensional case (Section 6). More generally, we show how to maintain an ϵ-*core-set* [1]. Although the space requirement of this method is dependent on a new variable R', defined as the ratio between diameter and the minimum width of a consecutive triple of points, we show such a dependence is necessary in the sliding window model.

2 Diameter in One Dimension: The Previous Algorithm

In this section, we briefly review the previous technique of Feigenbaum et al. [10] for maintaining the diameter of a one-dimensional point set over a sliding window of size at most N, as the ideas here will be helpful later (in Sections 5 and 6). At any particular time we have a set of points (or numbers) P in the sliding window, where an arrival time is associated to each point. We say that p is newer than q if the arrival time of p is greater than the arrival time of q (older otherwise). We want to approximate the diameter $\Delta(P) = \max_{p,q\in P} ||p-q||$, where $||p-q||$ denotes the distance between two points p and q.

The basic approach is similar to the *logarithmic method* of Bentley and Saxe [5] (which was also used in some previous data stream algorithms, *e.g.*, [1]). The input is split into several clusters. Each cluster represents an interval of time and the size of each cluster is a power of two. In each cluster C, a small subset $N_C \subseteq C$ of points (called *representative points*) is recorded as an approximation of C and obeys the following rules:

1. For each cluster C, the exact location of the newest point o_C is recorded as the *center* of C.
2. Each new point p forms a cluster C of size one.
3. For each point p in cluster C there exists $q \in N_C$ such that both p and q are on one side of o_C, q is not older than p, and
$$\frac{1}{1+\epsilon}||o_C-q|| \leq ||o_C-p|| \leq (1+\epsilon)||o_C-q||; \qquad (1)$$
we say that q is a *representative* of p (or p is *rounded* to q).
4. When there are more than two clusters of size k, the two older clusters C_1 and C_2 are combined to form a cluster C of size $2k$ and a merge procedure is applied to form N_C from N_{C_1} and N_{C_2}.

To construct a subset N_C satisfying Inequality 1 from C, the basic strategy is as follows: let o be the newest point in C and let δ be the distance of the closest point in C from o. For each $i \geq 0$, find the newest point at distance between $(1+\alpha)^i\delta$ and $(1+\alpha)^{i+1}\delta$ (for an α to be determined later) and put this point in N_C.

To merge two clusters C_1 and C_2, the idea is to apply the above procedure to compute N_C not from C but from just the points in $N_{C_1} \cup N_{C_2}$ (ignoring details about what δ should be for N_C).

A problem that arises is that each merge may cause an additional additive error of $\alpha\Delta(P)$. In other words, p might be rounded to q and then after a merge, q might be rounded to another point r. To overcome this problem, the idea is to use $\alpha = \frac{\epsilon}{\log N}$. Since at most $\log N$ merges can happen to a point p, it always has a representative q with $||q-p|| \leq (\log N)\alpha\Delta(P) = \epsilon\Delta(P)$.

In each cluster C, we keep $O(\log_{1+\alpha} R) = O(\frac{1}{\epsilon}\log N \log R)$ points in N_C, and there are $O(\log N)$ clusters, so the space complexity of this method is about the space needed for $O(\frac{1}{\epsilon}\log^2 N \log R)$ points, i.e., about $O(\frac{1}{\epsilon}\log^3 N \log R)$ bits.

3 Diameter in One Dimension: The New Algorithm

We now present a more direct algorithm for one-dimensional diameter that avoids working with multiple clusters. To find the furthest pair in P, it is enough to find the largest and smallest numbers, and we do this by two similar data structures that maintain an approximate maximum and minimum of the points. An "approximate maximum" here refers to a $q \in P$ such that $||q-p|| \leq \epsilon\Delta(P)$ where $p \in P$ is the maximum point.

We now describe a data structure that maintains a subset of $O(\log_{1+\epsilon} R)$ points of P, the largest of which is an approximate maximum. The data structure supports insertion of a new point and deletion of the oldest point. The key idea is in the following definition:

Definition 1. *Let $Q = \langle q_1, q_2, \ldots, q_k\rangle$ be a subsequence of P such that $q_1 < q_2 < \cdots < q_k$. Let $pred_Q(p)$ be the maximum value in Q that is at most p and $succ_Q(p)$ be the minimum value that is at least p. We call Q a* summary sequence *of P if the following conditions hold:*

1. *The q_i's are in decreasing order of arrival time.*
2. *For all p, $pred_Q(p)$, if it exists, is not older than p.*
3. *For all p, either $||p - pred_Q(p)|| \leq \epsilon\Delta_p(P)$ or $succ_Q(p)$ is not older than p.*

Here, $\Delta_p(P)$ denotes the diameter of all points in P not older than p.

Let us see why summary sequences are important. First notice that upon insertion of a new point p, we can delete all points $b \in P$ that are not greater than p. This is because from now on p is a better approximation than b for the maximum value. So, condition 1 can be guaranteed. Conditions 2 and 3 ensure that $\text{pred}_Q(p)$ or $\text{succ}_Q(p)$ can be used as a "representative" of p.

Notice that the summary sequence is not unique. For example in Figure 1, from the summary sequence $a_1, a_2, \ldots, a_7$, we can get another summary sequence by deleting a_3, a_4, and a_5 while ensuring condition 3. The interesting question is to how to maintain small summary sequences.

Our algorithm to maintain a summary sequence Q is remarkably simple and is described completely in the pseudocode below. We assume that Q is stored in a doubly linked list.

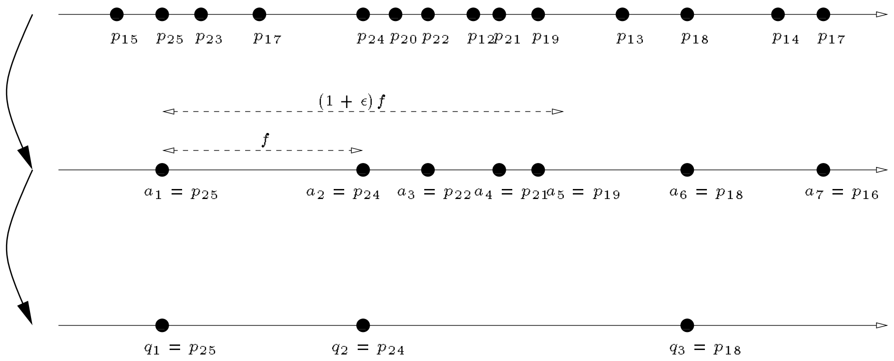

Fig. 1. An example of a summary sequence. (The index of each point is equal to its arrival time.)

Insert(p)**:** /* given a new point p */
1. Remove all points in Q that are less than p, and put p at the beginning.
2. After every $\frac{1}{\epsilon}\log R$ insertions, run Refine.

Refine:
1. Let q_1 and q_2 be the first and the second points in Q respectively.
2. Let $q := q_2$.
3. **While** q is not the last element of Q **do**
4. Let x and y be the elements before and after q in Q.
5. **If** $(1+\epsilon)||q_1 - x|| \geq ||q_1 - y||$ **then** remove q from Q.
6. Continue with q equal to y.

Delete(p)**:** /* given the oldest point p */
1. Just remove p from Q if $p \in Q$.

We prove the correctness of our algorithm:

Lemma 1. *After the refine algorithm, Q is still a summary sequence of P. Furthermore, $|Q| = O(\frac{1}{\epsilon}\log R)$.*

Proof: We show that conditions 1–3 remain valid each time a single point is removed from Q. This is obvious for condition 1. Regarding condition 2, just note that the new predecessor of a point is not older than the previous predecessor by condition 1.

We now consider condition 3. Let $p \in P$. Before the removal, we have either $||p - \mathrm{pred}_Q(p)|| \leq \epsilon\Delta_p(P)$ or $\mathrm{succ}_Q(p)$ is newer than p. Set $q = \mathrm{pred}_Q(p)$ in the former case, and $q = \mathrm{succ}_Q(p)$ in the latter. If q is not the point being removed, then the condition clearly remains true for the point p. Otherwise, by the design of the refine algorithm, the predecessor x and successor y of q must satisfy $(1+\epsilon)||q_1 - x|| \geq ||q_1 - y||$. This implies that $||p - x|| \leq ||y - x|| \leq \epsilon||x - q_1|| \leq \epsilon\Delta_p(P)$, because x and q_1 are newer than q by condition 1. Since x is the new predecessor of p, condition 3 remains true for the point p.

For the second part of the lemma, let $Q = \langle q_1, q_2, \ldots q_k \rangle$ after the refine algorithm. Then for each $1 < i < k-2$ we have $||q_1 - q_{i+2}|| > (1+\epsilon)||q_1 - q_i||$, because otherwise we would have removed q_{i+1}. Since $||q_1 - q_2||$ is at least the minimum non-zero distance $\delta(P)$, and $||q_1 - q_k||$ is at most the maximum distance $\Delta(P)$, we have $k \leq 2\log_{1+\epsilon} \frac{\Delta(P)}{\delta(P)} \leq 2\log_{1+\epsilon} R = O(\frac{1}{\epsilon}\log R)$. □

It is easy to check that conditions 1-3 remain true after line 1 of the insert algorithm, or after a deletion.

Theorem 1. *There is a data structure, storing $O(\frac{1}{\epsilon}\log R)$ points, that maintains a $(1+\epsilon)$-approximation of the diameter of a one-dimensional point set over a sliding window of variable size. Insertion of a new point takes $O(1)$ amortized time. Deletion of the oldest point can be done in $O(1)$ time.*

Proof: The number of points in the summary sequence is $O(\frac{1}{\epsilon}\log R)$ after each refine, and we do refine after each $\frac{1}{\epsilon}\log R$ insertions. Thus, the space complexity of this method is $O(\frac{1}{\epsilon}\log R)$. Upon insertion of a new point, we may remove $O(\frac{1}{\epsilon}\log R)$ points from the current summary sequence, but each point will be removed at most once. On the other hand, an execution of the refine algorithm requires $O(\frac{1}{\epsilon}\log R)$ time but is done once per $\frac{1}{\epsilon}\log R$ insertions. Thus, the amortized insertion time is $O(1)$. □

Remarks. The algorithm can be modified by standard techniques so that the worst-case insertion time is $O(1)$.

Besides being space-optimal (assuming coordinates have $O(\log N)$ bits), a desirable new feature of our data structure is that the size of the window need not be known in advance and can change dynamically (i.e., each insertion does not have to be followed by one deletion, or vice versa).

4 Diameter in Higher Fixed Dimensions

Feigenbaum et al. have extended their one-dimensional diameter algorithm to any fixed dimensions. They use the following well-known lemma:

Lemma 2. *There is a set L of $\Theta\left((\frac{1}{\epsilon})^{(d-1)/2}\right)$ lines, in $\mathbb{R}^d$ such that for each vector $x \in \mathbb{R}^d$, the angle between x and some $\ell \in L$ is at most $\arccos(\frac{1}{1+\epsilon})$.*

Such a set L can be constructed in $\Theta(|L|)$ time, for example, by a grid method [7]. The idea is to construct this set L and use the one-dimensional structure to maintain the diameter of the projection of the point set to each line $\ell \in L$.

Lemma 3. *Let L be the set of lines in Lemma 2. Assume that there is a black box $\mathcal{B}$ that, given a set of one-dimensional points, returns a $(1+\epsilon)$-approximation of its diameter. For a set of points P in $\mathbb{R}^d$, project each point to each line $\ell \in L$ and run $\mathcal{B}$ on it. The maximum of the returned values is a $(1+O(\epsilon))$-approximation of the diameter Δ of P.*

Proof: Let $p, q \in P$ be the furthest pair of points in P. Suppose the angle between line $\overleftrightarrow{pq}$ and $\ell \in L$ is at most $\arccos(\frac{1}{1+\epsilon})$. Let p' and q' be the projection of p and q on ℓ. Then $||p-q|| \leq (1+\epsilon)||p'-q'||$. On the other hand, the maximum returned value is at least $\frac{1}{1+\epsilon}||p'-q'|| \geq \frac{1}{(1+\epsilon)^2}\Delta$. □

Feigenbaum et al. did not observe the following problem: If we naively apply this projection approach, the spread of the one-dimensional point sets could be much bigger than the spread R of the actual point set P (because the closest-pair distance of the one-dimensional sets could be much smaller).

To fix this problem, we extend our one-dimensional approach using the above lemma but in a more creative way. (Our idea can also extend the result of Feigenbaum et al. to higher dimensions, but our data structure is more efficient.) We always keep the location of the two newest points p_1 and p_2 in the current window. If $Q^{(\ell)} = \langle q_1, q_2, \ldots, q_k \rangle$ is a summary sequence of projection of the given point set P on line ℓ, then before line 2 of the refine algorithm, we remove all q_i's that satisfies $||q_1 - q_i|| \leq \epsilon ||p_1 - p_2||$ (after the points are projected). These points are too close to q_1, and q_1 can serve as their representative. Condition 3 of summary sequences is still satisfied. Let $\delta(P)$ be the closest-pair distance of P. After the refine algorithm, $|Q|$ would then be bounded by $2\log_{1+\epsilon}\frac{\Delta(P)}{\epsilon\delta(P)} = O(\frac{1}{\epsilon}\log\frac{R}{\epsilon})$.

Theorem 2. *There is a data structure, storing $O((\frac{1}{\epsilon})^{(d+1)/2}\log\frac{R}{\epsilon})$ points, that maintains a $(1+\epsilon)$-approximation of the diameter of a point set in $\mathbb{R}^d$ over a sliding window of variable size. Insertion of a new point to the window takes $O((\frac{1}{\epsilon})^{(d-1)/2})$ amortized time, and deletion of the oldest point takes $O((\frac{1}{\epsilon})^{(d-1)/2})$ time.*

Proof: Each of the $O((\frac{1}{\epsilon})^{(d-1)/2})$ one-dimensional structures stores $O(\frac{1}{\epsilon}\log\frac{R}{\epsilon})$ points. An approximate diameter can be computed by Lemma 3. An insertion/deletion of a point may cause insertions/deletions in any of these one-dimensional structures. □

Remark. In the above theorem, we can also redefine R to be the ratio between the diameter and the minimum distance over all *consecutive* pairs (instead of all arbitrary pairs).

5 Diameter in Higher Non-Constant Dimensions

Although our algorithm can work in any fixed dimension, the complexity grows exponentially if d is not constant. If we allow a larger approximation factor, more precisely, $\sqrt{d}(1+\epsilon)$, we can still apply the projection approach to each of d main axes and get a structure that stores $O(d \log R)$ points for a fixed constant $\epsilon > 0$. To get an approximation factor independent of d, we suggest a different approach that requires larger (though sublinear) space. This approach in fact works for any metric space, where the distance function $d(\cdot,\cdot)$ satisfies the triangle inequality.

Lemma 4. *Let o', o, p, q be four points. If $d(o,p) \le \alpha d(o,q)$, then there exists $q' \in \{o,q\}$ such that $d(o',p) \le (2\alpha+1)d(o',q')$.*

Proof:

$$\begin{aligned} d(o',p) \le d(o',o) + d(o,p) \ &\le\ d(o,o') + \alpha d(o,q) \\ \le d(o,o') + \alpha[d(o,o') + d(o',q)] \ &\le\ (2\alpha+1)\max\{d(o',o), d(o',q)\}. \end{aligned}$$ □

For a sequence of points C in a metric space, let o_C be the newest point in C and let δ_C be the distance of the closest point in C to o_C. Define Q_C, the *summary set* of C, as follows: For each $i \ge 0$, put the newest point among points at distance between $(1+\epsilon)^i\delta_C$ and $(1+\epsilon)^{i+1}\delta_C$ into Q_C; also put o_C into Q_C.

Our algorithm proceeds as follows. We keep new points as they enter, and after every k insertions (for a parameter k to be determined later), we let C to be the set of k newest points (which we call a cluster) and replace C with Q_C. The main property regarding the summary set is that each point $p \in C$ has a representative $q \in Q_C$, not older than p, such that $\frac{1}{1+\epsilon}d(o_C,q) \le d(o_C,p) \le (1+\epsilon)d(o_C,q)$. Since $|Q_C| = O(\log_{1+\epsilon} R)$ for each C, our data structure keeps $O(k + \frac{n}{k}\log_{1+\epsilon} R)$ points.

To approximate the diameter of the whole window, we consider the center o_C of the newest cluster C and find the furthest point from o_C among all of the points kept in our data structure. (Note that the furthest point from o_C can be updated in $O(1)$ time after an insertion.) By the main property of summary sets and Lemma 4, if p is the furthest point from o_C in the whole window, then we have kept a point q not older than p such that $d(o_C,p) \le (2(1+\epsilon)+1)d(o_C,q)$. Since $d(o_C,p)$ is a 2-approximation of the diameter, we have a $6+O(\epsilon)$ approximation of the diameter.

Deletion of the oldest point is easy. Setting $k = \sqrt{N}$ yields the following result:

Theorem 3. *For any fixed $\epsilon > 0$, there is a data structure, storing $O(\sqrt{N}\log R)$ points, that maintains a $(6+\epsilon)$-approximation of the diameter over a sliding window of N points in a metric space. Insertion takes an $O(1)$ amortized number of distance computations. Deletion of the oldest point requires $O(\log R)$ time.*

Remark. The above method is similar to Feigenbaum et al.'s one-dimensional method (Section 2), but instead of $O(\log N)$ levels of merging, we use only one level. In fact, if we use multiple levels of merging, we can get better space complexity at the expense of a bigger approximation factor. More precisely, for any constant m, if we build a cluster for every $N^{\frac{1}{m+1}}$ points and merge the summary sets of every $N^{\frac{1}{m+1}}$ clusters of the same size together, then we can obtain a $(2^{m+2}-2+\epsilon)$-approximation algorithm by storing $O(N^{\frac{1}{m+1}}\log R)$ points.

6 Two-Dimensional Core-Sets

In this section, we consider geometric problems more difficult than the diameter under the sliding window model. Specifically, we show how to maintain an ϵ-*core-set* of a two-dimensional point set, as defined by Agarwal et al. [1]:

Definition 2. *Given a point set P in $\mathbb{R}^d$, the* extent measure *is defined as the function $\omega(P,x) = \max_{p,q\in P}(p-q)\cdot x$ for each $x \in \mathbb{R}^d$. An* ϵ-core-set *of P (under the extent measure) is a subset $E \subseteq P$ such that $\omega(E,x) \geq \frac{1}{1+\epsilon}\omega(P,x)$ for all x.*

Clearly, if S is an ϵ-core-set of a set P, then the width of S is a $(1+\epsilon)$-approximation of the width of P. Agarwal et al. [1] and Chan [7] gave efficient algorithms for constructing core-sets and applied them to develop approximation algorithms for various geometric optimization problems in both the traditional and data stream model.

Our algorithm in this section is for a fixed window size, and unfortunately its space bound depends on a new parameter in addition to ϵ, N, and R. This new parameter is the ratio between the diameter and the smallest width of each consecutive triple of points in the window and is denoted by R'. In Section 7 we show that the dependence on this new parameter is necessary for any algorithm that maintains an approximate width over a sliding window.

Our algorithm is a nontrivial combination of the diameter methods of Sections 2–4 with Chan's core-set method for data streams [7]. We start by using the clustering approach of Section 2 as in Feigenbaum et al. [10], *i.e.*, we store $O(\log N)$ clusters, and in each cluster C we attempt to keep a small subset of points N_C as the representative points of C.

The Subsets D_C: For each cluster C, we first maintain a set D_C of candidate points for a constant-factor approximation of the farthest point in C to the center o_C. This is done by following the diameter structure of Section 4, with say $\epsilon = 1/2$ (which maintains a constant number of summary sequences). Whenever two clusters C_1 and C_2 are combined to form C, we can compute D_C from D_{C_1} and D_{C_2} by combining the corresponding summary sequences of D_{C_1} and D_{C_2}. More precisely to combine two summary sequences both on the same line, we put the two sequences together and run the refine algorithm. When a point is deleted from C, we simply delete it from D_C.

Lemma 5. *Let p be an arbitrary point in a cluster C, then there exists a point $q \in D_C$ such that q is not older than p and $||o_C - p|| \leq 3||o_C - q||$.*

Proof: Let q and r be the two furthest points in D_C; we know $(1+\epsilon)||q-r|| \geq \Delta_p(C)$, where $\Delta_p(C)$ denotes the diameter of points in C that are not older than p. Then

$$||o_C - p|| \leq \Delta_p(C) \leq (1+\epsilon)||q-r|| \leq 2(1+\epsilon)\max\{||o_C - q||, ||o_C - r||\}.$$

The lemma follows as we have chosen $\epsilon = \frac{1}{2}$. □

The Representatives: For each $q \in D_C$, consider $\frac{6}{\alpha}$ lines perpendicular to $\overleftrightarrow{o_C q}$ at distances $0, \alpha||o_C - q||, 2\alpha||o_C - q||, \ldots, 3||o_C - q||$ from o_C, and call this set of lines L_q (the value of α will be specified later). For a point p and line ℓ, let $d(p, \ell)$ be the distance between p and ℓ. For $\ell \in L_q$, let C_ℓ be the set of points $p \in C$ such that $||o_C - p|| \le 3||o_C - q||$ and ℓ is the closest line to p in L_q. Let o be the newest point in C_ℓ. Let o_C, s_1, and s_2 be the newest, second newest, and third newest points C respectively and let w_C be the width of these three points.

Fix a point $q \in D_C$ and a line $\ell \in L_q$. Let p' denote the projection of a point p to ℓ and $C'_\ell = \{p' \mid p \in C_\ell\}$. For each $i \ge 0$, among the set of points whose projections are at distance between $(1+\alpha)^i \alpha w_C$ and $(1+\alpha)^{i+1} \alpha w_c$ from o', choose the newest point as the representative of these points. Also for points whose projection are at distance at most αw_C from o', choose o as their representative. With this selection, each point $p \in C_l$ has a representative v such that either $||v' - p'|| \le \alpha w_C$ or

$$\frac{1}{1+\alpha} \max\{||o'-v'||, ||o'-p'||\} \le ||o'-v'|| \le (1+\alpha) \min\{||o'-v'||, ||o'-p'||\}. \quad (2)$$

We let $v_q(p)$ denote such a representative v. Notice that here each point may have several representatives (possibly a different one for each $q \in D_C$). Notice that if $||p - o_C|| > 3||q - o_C||$, then $v_q(p)$ does not exist.

The above approach and the proof of the following lemma are similar to Chan's approach for data streams [7].

Lemma 6. *For a cluster C and a point $p \in C$ there exists a point $q \in D_C$ such that q is not older than p and for each vector x, $|(v_q(p) - p) \cdot x| \le O(\alpha)\omega_p(C, x)$, where $\omega_p(C, x) = \max_{a,b}(a - b) \cdot x$ and the maximum is over all $a, b \in C$ not older than p.*

Proof: By Lemma 5, pick a point $q \in D_C$ that is not older than p and $||o_C - p|| \le 3||o_C - q||$. Let $v = v_q(p)$ and o be the center of the set C_ℓ that contains p. Then for any $a \in C_\ell$,

$$|(a - a') \cdot x| \le |\frac{\alpha}{2}(o - q) \cdot x| \le \frac{\alpha}{2}\omega_p(C, x), \quad (3)$$

since neither q nor o is older than p. Applying this to p and v, we have

$$|(v - p) \cdot x| \le |(v - v') \cdot x| + |(v' - p') \cdot x| + |(p' - p) \cdot x| \le |(v' - p') \cdot x| + \alpha\omega_p(C, x).$$

If $||v' - p'|| \le \alpha w_C$, then we are done. Otherwise, by Inequality 2, $\frac{1}{1+\alpha}||o' - v'|| \le ||o' - p'|| \le (1 + \alpha)||o' - v'||$, implying that $|(v' - p') \cdot x| \le \alpha|(o' - p') \cdot x|$. Then

$$|(o' - p') \cdot x| \le |(o' - o) \cdot x| + |(o - p) \cdot x| + |(p - p') \cdot x| \le (1 + \alpha)\omega_p(C, x)$$

$$\Longrightarrow |(v - p) \cdot x| \le (\alpha(1 + \alpha) + \alpha)\omega_p(C, x). \qquad \square$$

The Subsets N_C: Whenever two clusters C_1 and C_2 are combined to form C, we construct N_C to be the set of all representatives of $N_{C_1} \cup N_{C_2}$. When a point is deleted from C, we simply delete it from N_C. By Lemma 6, each merging may cause an additive error of $O(\alpha)\omega_p(C, x)$. After a point p has experienced k merges, the additive error is $O(k\alpha)\omega_p(C, x)$. Since there could be $\log N$ merges, the final error will be $O(\alpha \log N)$, so to get an ϵ-core-set, we set $\alpha = \frac{\epsilon}{\log N}$.

Analysis: For the space requirement of our method, observe that $|D_C| = O(\log R)$, as in Theorem 2. For each $q \in D_C$, there are $O(\frac{1}{\alpha})$ lines, each of which has $O(\log_{1+\alpha} \frac{R'}{\alpha})$ representatives. So, $|N_C| = O(\frac{1}{\alpha} \log R \log_{1+\alpha} \frac{R'}{\alpha}) = O(\frac{1}{\alpha^2} \log R \log \frac{R'}{\alpha})$. Since $\alpha = \frac{\epsilon}{\log N}$, the number of points in each cluster is $O(\frac{1}{\epsilon^2} \log^2 N \log R \log \frac{R' \log N}{\epsilon})$. As there are $\log N$ clusters, we obtain the following theorem:

Theorem 4. *There is a data structure, storing $O(\frac{1}{\epsilon^2} \log^3 N \log R(\log R' + \log\log N + \log \frac{1}{\epsilon}))$ points, that maintains an ϵ-core-set of a two-dimensional point set over a sliding window of size N.*

Remark. Using core-sets, approximation algorithms can be developed for many geometric optimization problems, including the *smallest enclosing rectangle* and the *smallest enclosing circle* [1].

7 A Lower Bound for Width

We now prove a space lower bound for any algorithm that maintains an approximate width in the sliding window model. Our goal is to support Theorem 4 by showing that the dependence of the space lower bound on R' is necessary.

We use a proof technique similar to Feigenbaum et al.'s [10].

Theorem 5. *Let $\mathcal{A}$ be an algorithm that maintains a c-approximation of the width in the sliding window model. Given $R' \leq 2^{O(N^{1-\delta})}$ for some fixed $\delta > 0$, there are input streams for which the spread is bounded by a polynomial in N, the ratio of the diameter to the minimum width over all consecutive triples of points is bounded by $O(R')$, but $\mathcal{A}$ requires at least $\Omega(\log R' \log N)$ bits of space.*

Proof: Pick a non-increasing sequence $a_1 \geq a_2 \geq \cdots \geq a_N$ from the set $\{c^{-3}, c^{-6}, \ldots, c^{-3m}\}$ with $m = \frac{1}{3} \log_c R'$. Set $a_{N+1} = a_{N+2} = \cdots = 1/R'$. Consider the input stream of points $(-5 - i/N, 0), (i/N, a_i), (5 + i/N, 0)$ for $i = 1, 2, \ldots, 2N$. Since the diameter is $O(1)$ and the closest-pair distance is $\Omega(1/N)$, the spread is $O(N)$. The width of any consecutive triple is at least $1/R'$. If at any time we have a c-approximation to the width, we can reconstruct $a_1, \ldots, a_N$ after the point $(1, a_N)$ is read. Let M be the number of non-increasing sequences of size N chosen from a set of size m. The number of bits stored by the algorithm must therefore be at least

$$\log M = \log \binom{N + m - 1}{m - 1} = \Omega\left(m \log \frac{N}{m}\right) = \Omega\left(\log R' \log N\right).$$

$\square$

Remark. In contrast, if points have integer coordinates in the range $\{1, \ldots, R\}$, it is possible to replace the parameter R' with R in Theorem 4, because the width of any non-collinear triple of points is lower-bounded by $\Omega(1/R)$.

References

1. P. K. Agarwal, S. Har-Peled, and R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, to appear.
2. P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Computational Geometry: Theory and Applications*, 1:189–201, 1991.
3. P. K. Agarwal and M. Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete & Computational Geometry*, 16:317–337, 1996.
4. G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38:91–109, 2001.
5. J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformations. *Journal of Algorithms*, 1(4):301–358, 1980.
6. T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *International Journal on Computational Geometry and Applications*, 12:67–85, 2002.
7. T. M. Chan. Faster core-set constructions and data stream algorithms in fixed dimensions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 152–159, 2004.
8. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4(1):387–421, 1989.
9. C. A. Duncan, M. T. Goodrich, and E. A. Ramos. Efficient approximation and optimization algorithms for computational metrology. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 121–130, 1997.
10. J. Feigenbaum, S. Kannan, and J. Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica,* to appear; or as Tech. Report DCS/TR-1245, Yale University, `http://cs-www.cs.yale.edu/homes/jf/FKZ.ps`, 2002.
11. A. Goel, P. Indyk, and K. Varadarajan. Reductions among high dimensional proximity problems. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 769–778, 2001.
12. M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. Technical Report SRC-TN-1998-011, Hewlett Packard Laboratories, 1998.
13. J. Hershberger and S. Suri. Convex hulls and related problems in data streams. In *ACM SIGMOD/PODS Workshop on Management and Processing of Data Streams*, pages 148–168, 2003.
14. P. Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 539–545, 2003.
15. S. M. Muthukrishnan. Data streams: Algorithms and applications. Rutgers University Technical Report, `http://athos.rutgers.edu/~muthu/stream-1-1.ps`, 2003.
16. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.
17. E. A. Ramos. An optimal deterministic algorithm for computing the diameter of a three-dimensional point set. *Discrete & Computational Geometry*, 26:233–244, 2001.