# A Streaming Algorithm for 2-Center with Outliers in High Dimensions

Behnam Hatami* Hamid Zarrabi-Zadeh†

**Abstract**

We study the 2-center problem with outliers in high-dimensional data streams. Given a stream of points in arbitrary $d$ dimensions, the goal is to find two congruent balls of minimum radius covering all but $z$ points. We provide a $(1.8+\varepsilon)$-approximation streaming algorithm for the problem, improving upon the previous $(4+\varepsilon)$-approximation algorithm available for the problem. The space complexity and update time of our algorithm is $\text{poly}(d, z, \frac{1}{\varepsilon})$, independent of the size of the stream.

## 1 Introduction

The $k$-center problem—covering a set of points using $k$ congruent balls of minimum radius—is a fundamental problem, arising in many applications such as data mining, machine learning, statistics, and image processing. In real-world applications, where input data is often noisy, it is very important to consider outliers, as even a small number of outliers can greatly affect the quality of the solution. The $k$-center problem is particularly very sensitive to outliers, and even a constant number of outliers can increase the radius of the $k$-center unboundedly. Therefore, it is natural to consider the following generalization of the the $k$-center problem: given a set $P$ of $n$ points in arbitrary $d$ dimensions and a bound $z$ on the number of outliers, find $k$ congruent balls of minimum radius to cover at least $n-z$ points of $P$. See Figure 1 for an example.

In this paper, we focus on the *data stream* model of computation where only a single pass over input is allowed, and we have only a limited amount of working space available. This model is in particular useful for processing large data sets, as it does not require the entire data set to be stored in memory.

The Euclidean $k$-center problem has been extensively studied in the literature. If $k$ is part of the input, the problem in known to be NP-hard in two and more dimensions [10], and is even hard to approximate to within a factor better than 1.82, unless P = NP [9]. Factor-2 approximation algorithms are available for the problem in any dimension [11, 9]. For small $k$ and $d$, better

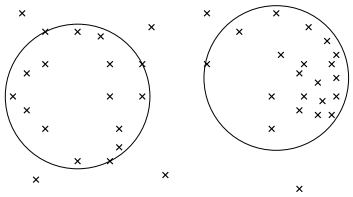

Figure 1: An example of 2-center with 6 outliers.

solutions are available. The 1-center problem in fixed dimensions is known to be LP-type and can be solved in $O(n)$ time [7]. For 2-center in the plane, the current best algorithm runs in $O(n \log^2 n \log^2 \log n)$ time [4].

For $k$-center with outliers, Charikar *et al.* [6] gave the first algorithm with an approximation factor of 3, which works in any dimension. Better results are known for small $k$ in the plane. The 1-center problem with $z$ outliers in the plane can be solved in $O(n \log n + z^3 n^{\varepsilon})$ time, for any $\varepsilon > 0$, using Matoušek's framework [14]. Agarwal [1] gave a randomized $O(nz^7 \log^3 z)$-time algorithm for 2-center with $z$ outliers in the plane.

In the streaming model, McCutchen *et al.* [15] and Guha [12] presented algorithms to maintain $(2+\varepsilon)$-approximation to the $k$-center problem in $O(\frac{kd}{\varepsilon} \log \frac{1}{\varepsilon})$ space. For $k=1$, a factor-$((1+\sqrt{3})/2)$ approximation was presented by Agarwal and Sharathkumar [2] in high dimensions, using $O(d)$ space. The approximation factor was later improved to 1.22 by Chan and Pathak [5]. For $k=2$, Kim and Ahn [13] have recently obtained a $(1.8+\varepsilon)$-approximation using $O(\frac{d}{\varepsilon})$ space and update time.

For $k$-center with $z$ outliers in the streaming model, McCutchen *et al.* [15] gave a $(4+\varepsilon)$-approximation algorithm using $O(\frac{zk}{\varepsilon})$ space. When dimension is fixed, a $(1+\varepsilon)$-approximation to 1-center with outliers can be maintained in $O(z/\varepsilon^{((d-1)/2)})$ space using the notion of robust $\varepsilon$-kernels [3, 16]. For 1-center with outliers in high dimensions, Zarrabi-Zadeh and Mukhopadhyay [17] gave a $(\sqrt{2}\alpha)$-approximation, where $\alpha$ is the approximation factor of the underlying algorithm for maintaining 1-center. Combined with the 1.22-approximation algorithm of Chan and Pathak [5], it yields an approximation factor of $(\sqrt{2} \times 1.22) < 1.73$ using $O(d^3 z)$ space and $\text{poly}(d, z)$ update time.

**Our result** In this paper, we study the 2-center problem with outliers in high dimensions. We present a

*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. bhatami@ce.sharif.edu

†Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. zarrabi@sharif.edu

| Problem | Approximation Factor | |
|---|---|---|
| | Without Outliers | With Outliers |
| 1-Center | 1.22 [5] | 1.73 [17] |
| 2-Center | $1.8+\varepsilon$ [13] | $1.8+\varepsilon$ [Here] |
| $k$-Center | $2+\varepsilon$ [12, 15] | $4+\varepsilon$ [15] |

Table 1: Summary of the streaming algorithms for $k$-center with and without outliers in high dimensions.

streaming algorithm for the problem that achieves an approximation factor of $1.8+\varepsilon$, for any $\varepsilon > 1$, using $\text{poly}(d, z, \frac{1}{\varepsilon})$ space and update time. This improves the current streaming algorithm available for the problem which has an approximation factor of $4+\varepsilon$. The approximation factor of our algorithm matches that of the best streaming algorithm for the 2-center problem without outliers. This is somewhat surprising, considering that the current best approximation factors for streaming $k$-center with and without outliers differ by a multiplicative factor of $\sqrt{2}$ for $k=1$, and by a factor of 2 for general $k$. See Table 1 for a comparison.

To obtain our result, we have used a combination of several ideas including parallelization, far/close ball separation, centerpoint theorem, and keeping a lower/upper bound on the radius and distance of the optimal balls. We have also utilized ideas used in [13] for the 2-center problem with no outliers. However, our problem is much harder here, as we not only need to find balls of minimum radius, but we also need to decide which subset of points to cluster. This is in particular more challenging in the streaming model, where we only have a single pass over the input, and we must decide on the fly which point is an outlier, and which one can be safely ignored as a non-outlier point, to comply with the working space restriction enforced by the model.

## 2 Preliminaries

Let $B(c,r)$ denote a ball of radius $r$ centered at $c$. We use $r(B)$ to denote the radius of a ball $B$. For two points $p$ and $q$, the distance between $p$ and $q$ is denoted by $\|pq\|$. Given two balls $B(c,r)$ and $B'(c',r')$, we define $\delta(B,B') = \max\{0, \|cc'\| - r - r'\}$ to be the *distance* between $B$ and $B'$. Two balls $B_1$ and $B_2$ are said to be *$\alpha$-separated*, if $\delta(B_1,B_2) \geqslant \alpha \cdot \max\{r(B_1), r(B_2)\}$.

Given an $n$-point set $P$ in $d$-dimensions, a point $c \in \mathbb{R}^d$ is called a *centerpoint* of $P$, if any halfspace containing $c$ contains at least $\lceil n/(d+1) \rceil$ points of $P$. It is well-known that any finite set of points in $d$-dimensional space has a centerpoint [8]. The following observation is a corollary of this fact.

**Observation 1** *Given a set $P$ of $k(d+1)$ points in $d$-dimensional space, the centerpoint of $P$ has the property that any convex object not covering the centerpoint, leaves at least $k$ points of $P$ uncovered.*

Given a point set $P$, the *$k$-furthest point from $p \in P$* is a point whose distance to $p$ is the $k$-th largest among all points in $P$. We assume the standard word-RAM model of computation. Each coordinate value takes a unit of space. Thus, a $d$-dimensional point takes $O(d)$ space, and basic operations on the points take $O(d)$ time.

## 3 A Simple Algorithm for 1-Center with Outliers

To warm up, we present a simple 2-approximation streaming algorithm for the 1-center problem with outliers. It utilized a parallelization technique that will be used extensively in the rest of the paper. The pseudocode is provided in Algorithm 1. The algorithm receives as input a stream of points, $P$, and the number of outliers, $z$. It assumes that the first point $p_1$ of the stream is non-outlier. We will show later how to remove this assumption. The algorithm returns a ball $B$ covering all but at most $z$ points of $P$.

**Algorithm 1** 1-CENTER$(P, z)$

```
1: c ← the first point in P
2: B ← B(c, 0)
3: Q ← ∅
4: for each p in P do
5:     if p ∉ B then
6:         insert p into Q
7:         if |Q| = z + 1 then
8:             q ← closest point to c in Q
9:             remove q from Q
10:            B ← B(c, ‖cq‖)
11: return B
```

**Theorem 1** *Algorithm 1 computes a 2-approximation to the 1-center problem with $z$ outliers, assuming that the first point of the stream is not outlier.*

**Proof.** Let $B^*(c^*, r^*)$ be the optimal solution, and $c$ be a non-outlier point in the optimal solution. Since $c$ is covered by $B^*$, for all points $p \in B^*$, we have $\|cp\| \leqslant \|cc^*\| + \|c^*p\| \leqslant 2r^*$. Among the $z+1$ points furthest from $c$, there is at least one point $q$ which is not outlier, and therefore, is contained in $B^*$ (see Figure 2). Thus, $\|cq\| \leqslant 2r^*$, and hence, the ball $B(c, \|cq\|)$ returned by Algorithm 1 is a 2-approximation. □

Algorithm 1 assumes that the first point of the stream is not outlier. To remove this assumption, we run $z+1$ instances of Algorithm 1 in parallel, each of which is given as input one of the first $z+1$ points of the stream, followed by the rest of the points. Clearly, there exists a point among the first $z+1$ points of $P$ which is not an

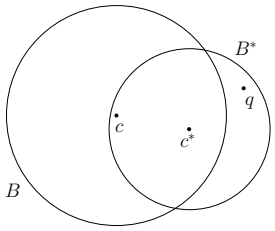


Figure 2: Proof of Theorem 1

outlier in the optimal solution. Therefore, the smallest ball among the $z+1$ balls computed in parallel is always within factor 2 of the optimal solution. Since the space complexity of Algorithm 1 for one instance is $O(zd)$, and its update time is $O(zd \log z)$, we get the following result.

**Theorem 2** *Given a stream of points in $d$ dimensions, we can maintain a 2-approximation to the 1-center with $z$ outliers in $O(z^2 d)$ space and $O(z^2 d \log z)$ update time.*

## 4 The 2-Center Problem with Outliers

In this section, we provide a $(1.8 + \varepsilon)$-approximation algorithm for the 2-center problem with outliers. In all algorithms presented in this section, we assume that the first point of the stream, $p_1$, is non-outlier. This assumption can be easily removed by considering $z + 1$ parallel instances of the algorithm, similar to what we did in Section 3.

Let $B_1^*$ and $B_2^*$ be the balls in an optimal solution to the 2-center problem with $z$ outliers on a point set $P$. We denote by $r^*$ the optimal radius, and by $\delta^*$ the distance between $B_1^*$ and $B_2^*$. To prove our main result, we distinguish between two cases. In Section 4.1, we address the case where $\delta^* > \alpha r^*$, for some constant $\alpha$ to be fixed later. (It will turn out that $\alpha = 16$ is a proper choice.) We then present in Section 4.2 our algorithm for the case of $\delta^* \leqslant \alpha r^*$.

### 4.1 The Case $\delta^* > \alpha r^*$

In this section, we present a 1.8-approximation algorithm for the case where optimal balls are separated by a distance greater than $\alpha r^*$. We start by two useful observations.

**Observation 2** *Let $B_1$ and $B_2$ be two congruent balls of radius $r$, with distance $\delta > \alpha r$. For any two points $p \in B_1$ and $q \in B_2$, we have $1 \leqslant \frac{\|pq\|}{\delta} < \frac{\alpha+4}{\alpha}$.*

**Proof.** The distance between $p$ and $q$ is at most $\delta + 4r$. Hence, $\frac{\|pq\|}{\delta} \leqslant 1 + \frac{4r}{\delta} < 1 + \frac{4}{\alpha}$. □

**Observation 3** *Let $B_1$ and $B_2$ be two disjoint balls of distance $\delta$, and let $B$ be an arbitrary ball of radius less than $\frac{\delta}{2}$. Then $B$ intersects at most one of $B_1$ and $B_2$.*

We next prove some properties regarding the optimal balls, $B_1^*$ and $B_2^*$.

**Lemma 3** *Let $B_1^*$ and $B_2^*$ be $\alpha$-separated, with $\alpha > 4$. If $p$ is a point in $B_1^*$, and $S$ is a $(z+1)$-subset of $P$ furthest from $p$, then $S \cap B_2^*$ is non-empty.*

**Proof.** Suppose by way of contradiction that $S \cap B_2^*$ is empty. Since $|S| = z + 1$, there is at least one point in $S$ which is not outlier, and hence, it is in $B_1^*$. Let $q$ be a point in $S \cap B_1^*$ furthest from $p$. Consider the ball $B(p, \|pq\|)$. For any point $s \in P \setminus S$, we have $\|ps\| \leqslant \|pq\|$, because $s \notin S$ and $q \in S$. Therefore, $B$ covers $P \setminus S$. Since $p, q \in B_1^*$, $\|pq\|$ is at most $2r^*$. Thus, by Observation 3, $B_2^* \cap B = \emptyset$. Therefore, $B_2^* \cap P = \emptyset$, and hence, $B_2^*$ is empty, which contradicts the optimality of the solution. □

**Lemma 4** *Let $p$ be a point in $B_1^*$, and $q$ be the $(z+1)$-furthest point from $p$. Then, $\delta^* > \frac{\alpha}{\alpha+4}\|pq\|$.*

**Proof.** By Lemma 3, there exists a point $q' \in B_2^*$ such that $\|pq'\| \geqslant \|pq\|$. Thus, by Observation 2, $\frac{\|pq\|}{\delta^*} \leqslant \frac{\|pq'\|}{\delta^*} < \frac{\alpha+4}{\alpha}$. □

**Lemma 5** *If $p \in B_1^*(c_1, r^*)$ and $q \in B_2^*(c_2, r^*)$, then $B_1^* \subset B(p, 2r^*)$ and $B_2^* \subset B(q, 2r^*)$, and hence, at most $z$ points of $P$ lie outside $B(p, 2r^*) \cup B(q, 2r^*)$.*

**Proof.** For an arbitrary point $p' \in B_1^*$, $\|pp'\| \leqslant \|pc_1\| + \|p'c_1\| \leqslant 2r^*$, and as a result, $B_1^* \subset B(p, 2r^*)$. Similarly, we have $B_2^* \subset B(q, 2r^*)$. Considering that at most $z$ points of $P$ are outlier, the proof is complete. □

**Lemma 6** *Let $S$ be a subset of $P$ of size at least $(d+1)(z+1)$, enclosed by a ball $B$ of radius less than $\delta^*/2$. Then the centerpoint $c_p$ of $S$ lies inside either $B_1^*$ or $B_2^*$.*

**Proof.** Not all points in $S$ can be outlier, because $(d+1)(z+1) > z$. Thus, by Observation 3, $B$ intersects exactly one of $B_1^*$ and $B_2^*$. Assume, w.l.o.g., that $B$ intersect $B_1^*$. Now, by Observation 1, if $c_p$ is not in $B_1^*$, then $z + 1$ points of $S$ remain uncovered by $B_1^*$, contradicting the fact that there at most $z$ outliers. □

**The Algorithm** We now describe our algorithm for handling the case $\delta^* > \alpha r^*$. At any time, our algorithm maintains a partition of $P$ into three disjoint subsets $B_1$, $B_2$, and Buffer. The first point $p_1$ is assumed, w.l.o.g, to be in $B_1^*$. (We have already assumed that $p_1$ is not outlier.) The algorithm tries to partition points in such a way that at the end, $B_1$ contains the whole $B_1^*$, and $B_2$ contains the whole $B_2^*$, with possibly some outliers being contained in $B_1$ and $B_2$. The algorithm sets $c_1 = p_1$ as the fixed center of $B_1$, and picks $c_2$ among the points processed so far as a candidate for being the center of

**Algorithm 2** 2-Center-Separated($P$)

```
1: c_1 ← p_1, r ← 0, δ ← 0
2: for each p ∈ P do
3:     if not (AddToB_1(p) or AddToB_2(p)) then
4:         add p to Buffer
5:         while |Buffer| > z do
6:             if |B_2| ⩾ (d+1)(z+1) then
7:                 B_1 ← B_1 ∪ B_2, B_2 ← ∅
8:             else if c_2 is set then
9:                 B_1 ← B_1 ∪ {c_2}
10:            T ← Buffer ∪ B_2 \ {c_2}
11:            B_2 ← ∅
12:            c_2 ← (z+1)-furthest point from c_1 in T
13:            r ← (2/α)‖c_1c_2‖
14:            for p ∈ T do
15:                AddToB_2(p)
16:            Buffer ← T \ B_2
```

$B_2$. Moreover, the algorithm maintains two values $\delta$ and $r$, where at any time, $\delta$ is a lower bound of $\delta^*$, and $r$ is an upper bound of $2r^*$ (under a certain condition).

Our algorithm is presented in Algorithm 2. For each input point $p \in P$, the algorithm first tries to add $p$ to either $B_1$ or $B_2$, using functions AddToB$_1$ and AddToB$_2$, respectively. If none of them fits, the point is added to Buffer. The function AddToB$_1$ adds a point $p$ to $B_1$ only if it is within distance $\delta$ of the center $c_1$. Similarly, AddToB$_2$ adds a point $p$ to $B_2$ only if it is within $r$-radius of $c_2$. The two functions also update the values of $\delta$ and $r$ whenever necessary, to maintain the invariants to be defined in Lemma 7.

**Algorithm 3** AddToB$_1$($p$)

```
1: if at least z + 1 points have been processed then
2:     q ← (z+1)-furthest point from c_1
3: else
4:     q ← c_1
5: δ ← (α/(α+4))‖c_1q‖
6: if p ∈ B(c_1, δ) then
7:     B_1 = B_1 ∪ {p}
8:     return true
9: return false
```

Whenever the buffer overflows (in line 5 of Algorithm 2), the algorithm takes one of the following actions depending on the size of $B_2$. If $|B_2| \geqslant (d+1)(z+1)$, then the points of $B_2$ are moved to $B_1$, and $B_2$ is reset. Otherwise, the old $c_2$ (if already set) is moved to $B_1$, and another point from $T = B_2 \cup \text{Buffer} \setminus \{c_2\}$ is picked as $c_2$. The while loop iterates at most $O(dz)$ times, because after the first iteration, we are sure that $T$ has at most $(d+1)(z+1)+z$ points, from which one point (i.e., $c_2$) is removed at each subsequent iteration.

For the sake of analysis, we maintain a "central point", denoted by $c_p$, which is defined as follows: if $|B_2| < (d+1)(z+1)$, then $c_p = c_2$, otherwise, $c_p$ is the centerpoint of the first $(d+1)(z+1)$ points currently in $B_2$.

**Algorithm 4** AddToB$_2$($p$)

```
1: if c_2 is set and p ∈ B(c_2, r) then
2:     B_2 ← B_2 ∪ {p}
3:     if |B_2| = (d+1)(z+1) then
4:         r ← (2 + 2/α) × r
5:         for p in Buffer do
6:             if p ∈ B(c_2, r) then
7:                 B_2 ← B_2 ∪ {p}
8:                 remove p from Buffer
9:     return true
10: return false
```

**Lemma 7** *The following invariants are maintained during the execution of the algorithm:*

*(a)* $\delta < \delta^*$

*(b)* $r \leqslant \delta/2$

*(c)* $B_1 \cap B_2^* = \emptyset$

*(d) if* $c_p \in B_2^*$, *then*

1. $2r^* \leqslant r$
2. $B_2 \cap B_1^* = \emptyset$
3. *all points in Buffer are outlier*

**Proof.** Invariant (a): At the beginning, $\delta = 0$, which clearly satisfies the invariant. After $z+1$ points of the stream is processed, function AddToB$_1$ starts updating $\delta$ to $\frac{\alpha}{\alpha+4}\|c_1q\|$, where $q$ is the $(z+1)$-furthest point from $c_1$ in the current stream. Now, since $c_1 \in B_1^*$, Lemma 4 implies that $\delta < \delta^*$.

Invariant (b): When $c_2$ is set by Algorithm 2, it is the $(z+1)$-furthest point from $c_1$ in a set $T \subseteq P$, and $r$ is set to $\frac{2}{\alpha}\|c_1c_2\|$. Let $q$ be the $(z+1)$-furthest point from $c_1$ in the stream at that moment. Then $\|c_1c_2\| \leqslant \|c_1q\|$. Assuming $\alpha \geqslant 16$, we have $\frac{2}{\alpha}\|c_1c_2\| \leqslant \frac{1}{6}\frac{\alpha\|c_1q\|}{(\alpha+4)} \leqslant \delta/6$, and hence,

$$r \leqslant (2+\frac{2}{\alpha})\frac{2}{\alpha}\|c_1c_2\| \leqslant 3 \times \frac{2}{\alpha}\|c_1c_2\| \leqslant \delta/2,$$

which means that the invariant holds, even after increasing $r$ by function AddToB$_2$.

Invariant (c): The proof is provided in Appendix C.

Invariant (d1): By Observation 2, if $c_1 \in B_1^*$ and $c_p \in B_2^*$, then $1 \leqslant \frac{\|c_1c_p\|}{\delta^*} \leqslant \frac{\|c_1c_p\|}{\alpha r^*}$, and as a result, $2r^* \leqslant \frac{2}{\alpha}\|c_1c_p\|$. If $|B_2| < (d+1)(z+1)$, then $c_p = c_2$, and by Algorithm 2, $r = \frac{2}{\alpha}\|c_1c_2\|$, and therefore, $2r^* \leqslant r$. If $|B_2| \geqslant (d+1)(z+1)$, then similar to invariant (b),

$$2r^* \leqslant \frac{2}{\alpha}\|c_1c_p\| \leqslant (1+\frac{2}{\alpha})\frac{2}{\alpha}\|c_1c_2\| \leqslant (2+\frac{2}{\alpha})\frac{2}{\alpha}\|c_1c_2\| = r.$$

Invariant (d2): By invariant (d1), if $c_p \in B_2^*$ then $2r^* \leqslant r \leqslant \delta/2$ and $c_p \in B_2$. Now, by invariant (a) and Observation 3, $B_2$ intersect only $B_2^*$, and hence, $B_2 \cap B_1^* = \emptyset$.

Invariant (d3): By invariants (c) and (d1), $2r^* \leqslant r \leqslant \delta/2 < \delta^*/2$. Therefore, by Lemma 5, all points outside $B_1 \cup B_2$ are outlier. □

**Theorem 8** *If $\delta^* > \alpha r^*$, a 1.8-approximation to the 2-center problem with $z$ outliers can be maintained in $O(d^3z^2)$ space and $\text{poly}(d, z)$ update/query time.*

**Proof.** Our algorithm for answering queries is provided in Appendix A. It uses the current partition $B_1$, $B_2$, and Buffer, to compute an optimal solution to 2-center with $z$ outliers. In the streaming model, we cannot afford keeping all the points of $B_1$ and $B_2$. Therefore, we maintain the sets $B_1$ and $B_2$ in a data structure that supports adding points, and gives a $\beta$-approximation to 1-center with $k$ outliers, for $k = 0, \ldots, z$. Moreover, we maintain a set $B_u = B_1 \cup B_2$ in a similar data structure. Note that these data structures do not need to maintain all the points. They only need to have a buffer of size $(d+1)(z+1)$ to keep the most recently added points.

To maintain $B_1$, $B_2$, and $B_u$, we use the streaming algorithm of [17, 5], which provides an approximation factor of $1.22 \times \sqrt{2} < 1.8$. The algorithm uses $O(d^3z)$ space and has $\text{poly}(d, z)$ update time. Since we need to run $z+1$ instances of Algorithm 2 in parallel, the space and update time are multiplied by a factor of $z$. □

### 4.2 The Case $\delta^* \leqslant \alpha r^*$

Our idea in this section is to carefully adopt the algorithm of Kim and Ahn [13], originally designed for maintaining an approximate 2-center. To avoid duplication, we just sketch the main steps of their algorithm, and explain our modifications to it. Kim and Ahn's algorithm, which we refer to as the KA algorithm, has 9 different states, shown in Figure 3. Depending on the points arrived so far, the algorithm is in one of the states. In each state, the algorithm keeps at most two balls as a candidate solution. A transition between the states occurs whenever a point not covered by any of the two balls arrive.

The algorithm starts at node 1, and proceed through the transition graph as points arrive. In some states, there is more than one state to follow, and the algorithm has no prior information which one is the correct choice. However, there are only three different paths to follow in the transition graph. Hence, we can easily run three instances of the algorithm in parallel, each of which follows one of the paths deterministically, to make sure that at any time, at least one of the instances is in a correct state.

Our modification is on the transition part. Points that are covered by the current solution can be safely ignored, as they do not cause any change in the current solution, and hence, they cause no transition. Only those points that lie outside the current solution are candidates for being outliers. Since the number of outliers in each state is unknown, we try all possible choices. The observation here is that the transition graph is a DAG of depth four. If $n_i$ $(1 \leqslant i \leqslant 4)$ represents the number of outliers in depth $i$, then it suffices to consider all tuples $(n_1, \ldots, n_4)$ such that $\sum_{i=1}^{4} n_i = z$, It is easy to verify that there are $O(z^3)$ such tuples.

The pseudocode of our algorithm is presented in Algorithm 5. For each possible choices of $n_1$ to $n_4$, and each of the three paths in the transition graph, numbered from 1 to 3, the algorithm keeps a candidate solution $(B_1, B_2)$ to the 2-center of non-outlier points, a parameter $j$ representing the current level in the transition graph, and four counters to keep track of number of outliers seen so far at each level.

The algorithm starts with $B_1 = B(p_1, r)$ and $B_2 = \emptyset$, which corresponds to Case 1 of the KA algorithm. For each new point $p$, we first check if it is contained in the current solution. If so, then we are done. Otherwise, if the number of outliers seen in the current level has not yet reached $n_j$, we consider $p$ as an outlier and pro-

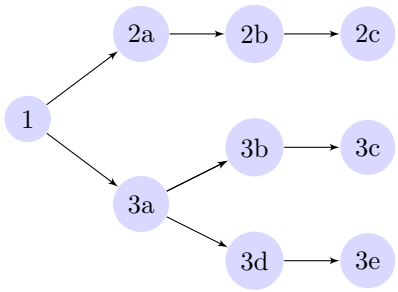


Figure 3: State diagram of the KA algorithm. Labels are taken from [13].

**Algorithm 5** 2-CENTER-CLOSE$(P, z, r)$

```
1: solutions ← {}
2: for each (n1, n2, n3, n4) such that ∑ ni = z do
3:     for each π ∈ {1, 2, 3} do
4:         counter_i ← 0, for i = 1, ..., 4
5:         B1 ← B(p1, r), B2 ← ∅
6:         j ← 1    ▷ j represents current level
7:         for each p ∈ P do
8:             if p ∉ B1 ∪ B2 then
9:                 counter_j ← counter_j + 1
10:                if counter_j > n_j then
11:                    j ← j + 1
12:                (B1, B2) ← KA.INSERT(p, π)
13:        if j ⩽ 4 then
14:            add max{r(B1), r(B2)} to solutions
15: return min{solutions}
```

ceed. Otherwise, we go to the next level, and update the current candidate solution, $(B_1, B_2)$, using the KA algorithm. We give the transition path $\pi$ along with the point $p$ to the KA algorithm to help it deterministically decide which state to choose as the next one.

After all points in $P$ are processed, if we are in one of the four states in the current path, then the obtained solution is added to the feasible solutions. Otherwise, the solution is not feasible, and is abandoned as in the KA algorithm. Finally, we return the best solution among all computed feasible solutions. Kim and Ahn [13] proved that in all feasible solutions computed this way, the larger ball among $B_1$ and $B_2$ has radius at most $3/2r$, provided $\delta^* \leqslant \alpha r^*$. (Their proof is stated for $\alpha = 2$, but can be extended to any $\alpha \geqslant 2$.) Assuming that we have a good estimate $r$ satisfying $1.2r^* \leqslant r < (1.2 + 2\varepsilon/3)r^*$, we get the following.

**Theorem 9** *For $1.2r^* \leqslant r < (1.2 + \frac{2}{3}\varepsilon)r^*$ and $\delta^* \leqslant \alpha r^*$, Algorithm 5 computes a $(1.8+\varepsilon)$-approximation to the 2-center with $z$ outliers in $O(dz^3)$ space and $O(dz^3)$ update time, assuming that the first point of the stream is not outlier.*

As shown in Appendix B, a desired estimate for $r$ can be obtained by running $O(1/\varepsilon)$ instances of Algorithm 5 in parallel. Adding another level of parallelization to remove the assumption of $p_1$ being a non-outlier, we get the following.

**Theorem 10** *If $\delta^* \leqslant \alpha r^*$, a $(1.8+\varepsilon)$-approximation to the 2-center problem with $z$ outliers can be maintained in $O(\frac{dz^4}{\varepsilon})$ space and $O(\frac{dz^4}{\varepsilon})$ update/query time.*

Theorems 8 and 10 together yield the following main result of the paper.

**Theorem 11** *Given a stream of points in $d$ dimensions, we can maintain a $(1.8+\varepsilon)$-approximation to the 2-center problem with $z$ outliers using $O(dz^2(d^2+z^2/\varepsilon))$ space and $\mathrm{poly}(d, z, \frac{1}{\varepsilon})$ update/query time.*

## 5 Conclusions

In this paper, we presented a $(1.8 + \varepsilon)$-approximation streaming algorithm for 2-center problem with outliers in Euclidean space. It improves the previous $(4 + \varepsilon)$-approximation algorithm available for the problem due to McCutchen and Khuller [15]. Finding better approximation factor or space complexity is an interesting problem that remains open. It is also interesting to see if the ideas in this paper can be extended to the $k$-center problem with outliers in the data stream model, even for small values of $k \geqslant 3$.

**Acknowledgement** The authors would like to thank Kiana Ehsani and Sahand Mozaffari for their thoughtful discussions, and for their very helpful comments.

## References

[1] P. K. Agarwal and J. M. Phillips. An efficient algorithm for 2d Euclidean 2-center with outliers. In *Proc. 16th Annu. European Sympos. Algorithms*, pages 64–75. 2008.

[2] P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. In *Proc. 21st ACM-SIAM Sympos. Discrete Algorithms*, pages 1481–1489, 2010.

[3] P. K. Agarwal and H. Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 1–10, 2007.

[4] T. M. Chan. More planar two-center algorithms. *Comput. Geom. Theory Appl.*, 13(3):189–198, 1999.

[5] T. M. Chan and V. Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Comput. Geom. Theory Appl.*, 47(2):240–247, 2014.

[6] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 642–651, 2001.

[7] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *J. Algorithms*, 21(3):579–597, 1996.

[8] L. Danzer, B. Gruenbaum, and V. Klee. Helly's theorem and its relatives. In *Proc. Symposia in Pure Mathematics 7*, pages 101–180, 1963.

[9] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 434–444, 1988.

[10] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Inform. Process. Lett.*, 12(3):133–137, 1981.

[11] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[12] S. Guha. Tight results for clustering and summarizing data streams. In *Proc. 12th Internat. Conf. Database Theory*, pages 268–275, 2009.

[13] S.-S. Kim and H.-K. Ahn. An improved data stream algorithm for clustering. In *Proc. 11th Latin American Theoret. Inform. Sympos.*, pages 273–284. 2014.

[14] J. Matoušek. On geometric optimization with few violated constraints. *Discrete Comput. Geom.*, 14(1):365–384, 1995.

[15] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 165–178. 2008.

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

[17] H. Zarrabi-Zadeh and A. Mukhopadhyay. Streaming 1-center with outliers in high dimensions. In *Proc. 21st Canad. Conf. Computat. Geom.*, pages 83–86, 2009.

## A Answering Queries

In the following, we show how the information maintained by Algorithm 2 can be used to answer queries of this kind: find two $\alpha$-separated congruent balls of minimum radius to cover all but at most $z$ points of the stream processed so far.

Our query algorithm is presented in Algorithm 6. The idea behind the algorithm is as follows. By our initial assumption about $p_1$ and by invariants (c) and (d), if $c_p \in B_2^*$, then we know that $B_1$ completely contains $B_1^*$, and $B_2$ completely contains $B_2^*$. However, it might be the case that our assumption about $c_p$ was incorrect, and therefore, $B_1$ (resp., $B_2$) may not completely contain $B_1^*$ (resp., $B_2^*$). To overcome this issue, we try all possible candidates for $c_2$ (which in turn, determines $c_p$), and compute, for each resulting partition of $P$ into $B_1$, $B_2$, and Buffer, the best solution for 2-center with $z$ outliers using the MinCover function presented in Algorithm 7.

**Algorithm 6** Query

```
solutions ← {}
candidates ← B_2 ∪ Buffer
if |B_2| ⩾ (d+1)(z+1) then
    candidates ← {c_2} ∪ Buffer
    B_1 ← B_1 ∪ B_2
δ_0 = δ
for c ∈ candidates do
    r ← (2/α)‖c_1 c‖
    B'_1 ← B_1, δ ← max{δ_0, r}
    B'_2 ← ∅, Buffer' ← ∅
    for p ∈ candidates do
        if not (AddToB'_1(p) or AddToB'_2(p)) then
            add p to Buffer'
    add MinCover(B'_1, B'_2, Buffer') to solutions
return min{solutions}
```

**Algorithm 7** MinCover$(B_1, B_2, \text{Buffer})$

```
solutions ← {}
for k ← 0, ..., (z − |Buffer|) do
    r_1 ← 1-Center(B_1, k)
    r_2 ← 1-Center(B_2, z − |Buffer| − k)
    add max{r_1, r_2} to solutions
return min{solutions}
```

Let $C$ denote the set of candidates for $c_2$. By invariant (c), we know that $B_1 \cap B_2^* = \emptyset$. Therefore, there exists a point in $(B_2 \cup \text{Buffer}) \cap B_2^*$, and hence, $C$ is $(B_2 \cup \text{Buffer}) \cap B_2^*$ in general. However, when $|B_2| \geqslant (d+1)(z+1)$, we will show in the following lemma that $(\{c_2\} \cup \text{Buffer}) \cap B_2^* \neq \emptyset$. Therefore, if $|B_2| \geqslant (d+1)(z+1)$, we only need to consider $\{c_2\} \cup \text{Buffer}$ as candidates for $C$.

**Lemma 12** *At any time, if $|B_2| \geqslant (d+1)(z+1)$ and $c_p \notin B_2^*$, then $B_2 \cap B_2^* = \emptyset$.*

**Proof.** By invariants (a) and (b), we know that $r \leqslant \delta/2 < \delta^*/2$. By Lemma 6, $c_p \in B_1^* \cap B_2^*$. Since $c_p \notin B_2^*$, we have $c_p \in B_1^*$. On the other hand, by Observation 3, $B_2$ intersect at most one of $B_1^*$ and $B_2^*$. Therefore, $B_2 \cap B_2^* = \emptyset$. □

Our query algorithm works as follows. For each candidate point $c \in C$, Algorithm 6 constructs $B'_1(c_1, \max\{\delta, \frac{2}{\alpha}\|c_1 c\|\})$ and $B'_2(c, \frac{2}{\alpha}\|c_1 c\|)$. If the candidate $c$ equals the current $c_2$, then we have $B_1 = B'_1$. Since $\frac{2}{\alpha}\|c_1 c_2\| \leqslant r \leqslant \delta/2$ by invariant (b), and $B'_2 \subset B_2$, we do not need to construct any new set. For $c \neq c_2$, we know that $B_1 \subset B'_1$, and hence, we only need to see which points in Buffer $\cap B_2$ are inside $B'_1$. When $|B_2| \geqslant (d+1)(z+1)$ and $c \neq c_2$, then it means that $c_p \notin B_2^*$. Therefore, by Lemma 12, $B_2$ can be added to $B'_1$ without violating invariant (c). So in this case, we just need to see which points of Buffer must be added to $B'_1$. Algorithm 6 uses functions AddToB$'_1$ and AddToB$'_2$ for adding a point to $B'_1$ and $B'_2$ respectively. These functions are the same as AddToB$_1$ and AddToB$_2$, with the only exception that they add points to $B'_i$ instead of $B_i$, for $i = 1, 2$.

Since Algorithm 6 considers all valid candidates for $c$, at least for one $c^* \in C$, we have $c^* \in B_2^*$. We denote the corresponding $B'_1$ and $B'_2$ by $B''_1$ and $B''_2$. Since by Observation 2, $1 \leqslant \frac{\|c_1 c^*\|}{\alpha^*} < \frac{\|c_1 c^*\|}{\alpha r^*}$, we have $2r^* \leqslant \frac{2}{\alpha}\|c_1 c^*\|$, and hence by Lemma 5, $B_1^* \subset B''_1$ and $B_2^* \subset B''_2$. On the other hand, since the distance of the new points added to $B''_1$ is less than $\|c_1 p\| \leqslant \max\{\delta, \frac{2}{\alpha}\|c_1 c^*\|\}$, we have by invariant (c) that $B''_1 \cap B_2^* = \emptyset$. As a result, $B''_1$ (resp., $B''_2$) completely covers $B_1^*$ (resp., $B_2^*$), and the points in Buffer are all outliers. The only unknown part is that Algorithm 6 does not know how many outliers are in $B''_1$ and $B''_2$. Therefore, Algorithm 7 tries all possible cases and choose the one with the minimum radius.

## B Estimating $r$

In this section, we show how to obtain a value $r$, such that $1.2r^* \leqslant r < (1.2 + 2\varepsilon/3)r^*$. The following lemma provides the main ingredient.

**Lemma 13** *Given a point set $P$ in $\mathbb{R}^d$, an optimal solution to the 1-center problem with $z$ outliers on $P$ gives a $(2+\frac{\alpha}{2})$-approximation for the 2-center with $z$ outliers on $P$, provided that $\delta^* \leqslant \alpha r^*$.*

**Proof.** Let $r_1^*$ and $r^*$ be the optimal radii for the 1-center and 2-center problems with $z$ outliers on $P$, respectively. It is clear that $r^* \leqslant r_1^*$, because any feasible solution $B^*$ for 1-center with $z$ outliers yields a feasible solution $(B^*, B^*)$ for 2-center with $z$ outliers. Now,

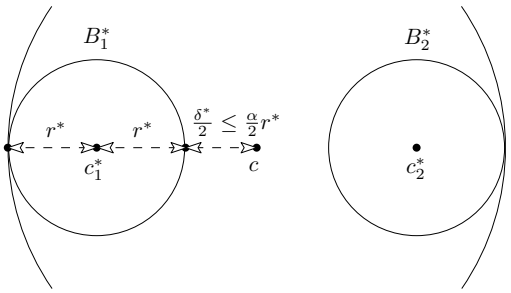


Figure 4: Proof of Lemma 13

suppose that $B_1^*(c_1^*, r^*)$ and $B_2^*(c_2^*, r^*)$ are the balls in an optimal solution for the 2-center problem with $z$ outliers. Let $c$ be the midpoint of the segment connecting $c_1^*$ to $c_2^*$ (see Figure 4). Clearly, $B\left(c, \frac{\delta}{2} + 2r^*\right)$ covers both $B_1^*$ and $B_2^*$. Therefore, it is a feasible solution for the 1-center problem with $z$ outliers. Hence, $r_1^* \leqslant \left(2 + \frac{\alpha}{2}\right) r^*$. □

The following is a direct corollary of Lemma 13 and Theorem 1.

**Corollary 14** *If $\delta^* \leqslant \alpha r^*$, Algorithm 1 computes a $(4+\alpha)$-approximation to $r^*$.*

We use Algorithm 1 to find an estimate for $r$. Let $r_i$ be the radius calculated by Algorithm 1 after receiving the $i$-th point, $p_i$. Clearly, the sequence of $r_i$'s is increasing. Let $k$ be an integer such that $2^{k-1} \leqslant r_i \leqslant 2^k$, and set $\ell_i = 2^k$. (If $r_i = 0$, we set $\ell_i = 0$.) Obviously, $\ell_i \leqslant 2r_i$, and hence, by Corollary 14, $\ell_i$ is a $(8+2\alpha)$-approximation to $r^*$.

We divide the interval $(0, 1.2\ell_i]$ into $m = \lceil 1.2(3\alpha+12)/\varepsilon \rceil$ equal segments, each of length $t_i = 1.2\ell_i/m$. Clearly, $t_i \leqslant (2\varepsilon/3)r^*$. Therefore, in the set $R_i = \{j \times t_i \,|\, j = 1, \ldots, m\}$, there is at least one value $r$ for which the inequality $1.2r^* \leqslant r \leqslant (1.2 + \frac{2\varepsilon}{3})r^*$ holds.

We run $m$ instances of Algorithm 5 for each value $r \in R_i$ in parallel. Whenever a new point $p_i$ is added, if $\ell_i = \ell_{i-1}$, then $R_i = R_{i-1}$, and the new point is inserted to all parallel instances. If $\ell_i > \ell_{i-1}$, then the set $R_i$ has two types of values. Those values in $R_i$ which are less than $1.2\ell_i$ are also present in $L_{i-1}$, because $t_i/t_{i-1}$ is a positive power of 2. For these values, we continue executing the corresponding instance. If a value $r \in R_i$ is not present in $R_{i-1}$, then we have $r \geqslant 1.2\ell_{i-1} \geqslant \ell_{i-1}$. Since those points not lying in the candidate solution are saved in the buffer of Algorithm 1 (which has size at most $z$), all non-outlier points of this algorithm lie in the candidate balls of Algorithm 5 which has center $p_1$ and radius at most $l_{i-1}$. These outliers have been stored in a buffer. Since Algorithm 5 maintains two balls with radius at least $r$, one of which (say $B_1$) is centered at $p_1$, then all non-outlier points of Algorithm 1 are in $B_1$, and hence, they do not make any transition in the states of Algorithm 5. Therefore, for any new value $r$, it suffices to execute Algorithm 5 with only the outlier points in the buffer of Algorithm 1.

## C Proof of Invariant (c)

Here, we provide a proof for Invariant (c). The following technical claim will be used in our proof.

**Claim 1** *If $c_2$ is set, then $B(c_p, \frac{2}{\alpha}\|c_1c_p\|) \subseteq B_2(c_2, r)$.*

**Proof.** If $|B_2| < (d+1)(z+1)$, then $c_p = c_2$ and $r = \frac{2}{\alpha}\|c_1c_2\|$, and hence, $B_2 = B(c_p, \frac{2}{\alpha}\|c_1c_p\|)$. When the size of $B_2$ reaches $(d+1)(z+1)$, the central point $c_p$ moves to the centerpoint of $B_2$, and $r$ is increased by a factor of $(2 + \frac{2}{\alpha})$. Because the centerpoint of $B_2$ lies in $B_2$, then $c_p \in B(c_2, \frac{2}{\alpha}\|c_1c_2\|)$. Thus, if $|B_2| \geqslant (d+1)(z+1)$ then $\|c_2c_p\| \leqslant \frac{2}{\alpha}\|c_1c_2\|$, and therefore,

$$\frac{2}{\alpha}\|c_1c_p\| \leqslant \frac{2(\|c_1c_2\| + \|c_2c_p\|)}{\alpha} \leqslant \frac{2}{\alpha}\|c_1c_2\|(1 + \frac{2}{\alpha}).$$

Hence, $B(c_p, \frac{2}{\alpha}\|c_1c_p\|) \subseteq B_2(c_2, \frac{2}{\alpha}\|c_1c_2\|(2 + \frac{2}{\alpha}))$. □

Now, we prove Invariant (c), which states $B_1 \cap B_2^* = \emptyset$.

**Proof.** A point $p$ can be added to $B_1$ in two cases. The first case is in function AddToB$_1$, where the point is added to $B_1$ only if it is within distance $\delta$ of the center $c_1$, which by invariant (a), guaranties $\|pc_1\| < \delta^*$. Therefore, $p \notin B_2^*$ in this case.

The second case is in Algorithm 2, when the buffer overflows and $B_2$ is non-empty. The algorithm takes one of the following actions depending on the size of $B_2$. If $|B_2| < (d+1)(z+1)$, then $c_2 = c_p$. Algorithm 2 adds $c_2$ to $B_1$. Suppose by way of contradiction that $c_p \in B_2^*$. Then, by invariants (b) and (d1), $2r^* \leqslant r \leqslant \delta/2 < \delta^*/2$. Therefore, by Lemma 5, there must be at most $z$ points outside $B_1 \cup B_2$, which contradicts the overflow of the buffer. If $|B_2| \geqslant (d+1)(z+1)$, then $c_p$ is the centerpoint of the first $(d+1)(z+1)$ points currently in $B_2$. In this case, we add all points of $B_2$ to $B_1$. By invariant (b), $r \leqslant \delta/2 < \delta^*/2$. Therefore, By Lemma 6, $c_p \in B_1^*$ or $c_p \in B_2^*$. Suppose by way of contradiction that $c_p \in B_2^*$. In this case, by invariant (d1) and Claim 1, $B_2$ covers $B(c_p, \frac{2}{\alpha}\|c_1c_p\|)$ and $2r^* \leqslant r$. Therefore, Similar to the previous part, it contradicts the overflow of the buffer. □