Streaming Algorithms for
Smallest Intersecting Ball of Disjoint Balls

Wanbin Son and Peyman Afshani

MADALGO*, Department of Computer Science, Aarhus University, Denmark
{wson,peyman}@cs.au.dk

Abstract. In this paper, we propose streaming algorithms for approximating the smallest intersecting ball of a set of disjoint balls in $\mathbb{R}^d$. This problem is a generalization of the 1-center problem, one of the most fundamental problems in computational geometry. We consider the single-pass streaming model; only one-pass over the input stream is allowed and a limited amount of information can be stored in memory. We introduce three approximation algorithms: one is an algorithm for the problem in arbitrarily dimensions, but in the other two we assume d is a constant. The first algorithm guarantees a $(2+\sqrt{2}+\varepsilon^*)$-factor approximation using $O(d^2)$ space and $O(d)$ update time where ε^* is an arbitrarily small positive constant. The second algorithm guarantees an approximation factor 3 using $O(1)$ space and $O(1)$ update time (assuming constant d). The third one is a $(1 + \varepsilon)$-approximation algorithm that uses $O(1/\varepsilon^d)$ space and $O(1/\varepsilon^{(d-1)/2})$ amortized update time. They are the first approximation algorithms for the problem, and also the first results in the streaming model.

1 Introduction

Given a set D of n pairwise interior-disjoint balls in $\mathbb{R}^d$, $n > d$, we consider the problem of finding a center that minimizes the maximum distance between the center and a ball in D. The distance between a point p and a ball b centered at c of radius r is defined by $dist(p, b) = \max\{0, |pc| - r\}$ where $|\cdot|$ is a distance between two points. The problem can be also formulated as finding the smallest ball that intersects all the input balls, and hence we call it "the smallest intersecting ball of disjoint balls (SIBB) problem". Observe that if the input balls are points, which we call "the smallest enclosing ball of points (SEBP) problem", then we have an instance of the 1-center problem. The 1-center problem is a fundamental problem in computational geometry, specially in the area of facility location problems [1]. So in our problem we model facilities that can be "relocated" up to a fixed distance. Finally, as an additional motivation, we can view each ball as an uncertain point [2] and thus a solution for the SIBB problem would imply a lower bound for the 1-center problem in a set of uncertain points.

Several properties of the SIBB problem are same as those of the SEBP problem. Both problems are LP-type problems with combinatorial dimension $d + 1$ [3], so the problem can be solved by a generic algorithm for an LP-type problem in the static setting. Because of the similarity between the SEBP problem and the SIBB problem, few studies [1,2,4] consider the SIBB problem.
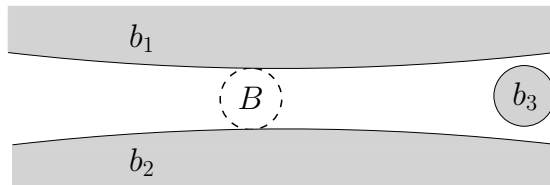
Fig. 1. Three input balls b_1 b_2 and b_3, and the smallest ball B that intersects b_1 and b_2.

In this paper, we are interested in the problem in the *single-pass streaming model*; only one-pass over the input stream is allowed and only a limited amount of information can be stored. The streaming model is attractive both in theory and in practice due to massive increase in the volume of data over the last decades, a trend that is most likely to continue. We assume that the memory size is much smaller than the size of input data in the streaming model, so it is important to develop an algorithm in which the space complexity does not depend on the size of input data.

In the streaming model, the similarities between the SIBB and SEBP problems break down as it turns out that a set of balls is much more difficult to process than a set of points. For example, let us consider a factor 1.5-approximation algorithm for the SEBP problem [5] which works as follows: For the first two input points, the algorithm computes the smallest enclosing ball B for them. For each next input point p_i, the algorithm updates B to be the smallest ball that contains B and p_i. For the SEBP problem, the algorithm gives the correct approximation factor. The obvious extension of this algorithm to the SIBB problem would be as follows: For the first two input balls b_1 and b_2, compute the smallest intersecting ball B for them. For each next input ball b_i, compute the smallest ball that contains B and intersects b_i. This algorithm gives a solution that intersects all the input balls, but it does not guarantee any approximation factor. In fact, as as Figure 1 shows, the approximation factor could be arbitrarily large: the first two input balls are b_1 and b_2, and B is the smallest intersecting ball for them. The third input ball on the stream is B_3 and the smallest ball that contains B and intersects b_3 can be arbitrarily larger than the optimal solution if b_3 is located far way from B. The reason why the algorithm does not work properly is that B does not "keep" enough information about b_1 and b_2. Because of similar reasons, the other approximation algorithms [5,6,7] for the SEBP problem also do not work for the SIBB problem.

Previous work in the static setting. Matoušek et al. [3] showed that the smallest intersecting ball of convex objects problem is an LP-type problem, so it can be solved in $O(n)$ time in fixed dimensions. Löffler and Kreveld [2] considered the the smallest intersecting ball of balls problem as the 1-center problem for imprecise points. They mentioned that the problem is an LP-type problem. Mordukhovich et al. [1] described sufficient conditions for the existence and uniqueness of a solution for the problem. In the plane, Ahn et al. [4] proposed an algorithm to compute the smallest two congruent disks that intersect all the input disks in $O(n^2 \log^4 n \log \log n)$ time.

For the SEBP problem, it is known that the problem is an LP-type problem, so it can be solved by an LP-type framework in linear time in fixed constant dimensions [3]. While the LP-type framework gives an exact solution, it is not attractive when d may be large because a hidden constant in the time complexity of the LP-type framework has exponential dependency on d. In high dimensions, Bâdoiu and Clarkson [8] presented a $(1 + \varepsilon)$-approximation algorithm that computes a solution in $O(nd/\varepsilon + (1/\varepsilon)^5)$ time.

The k-center problem is NP-hard if k is a part of input [9], so studies have been focused on the problem for small k [10,11] or developing approximation algorithms [12,13].

Previous work on data streams. To the best of our knowledge, our work contains the first approximation algorithms for the SIBB problem, and also the first results in the streaming model.

The SEBP problem, however, has been studied extensively in the streaming model. Zarrabi-Zadeh and Chan [5] showed a 1.5 approximation algorithm which uses the minimum amount of storage. Agarwal and Sharathkumar [6] presented a $((1 + \sqrt{3})/2 + \varepsilon)$-approximation algorithm using $O(d/\varepsilon^3 \log{(1/\varepsilon)})$ space, and Chan and Pathak [7] proved that the algorithm has approximation factor 1.22. Agarwal and Sharathkumar [6] also showed that any algorithm in the single-pass stream model that uses space polynomially bounded in d cannot achieve an approximation factor less than $(1 + \sqrt{2})/2 > 1.207$. In fixed dimensions, a $(1 + \varepsilon)$-approximation algorithm can be derived using $O(1/\varepsilon^{(d-1)/2})$ space and $O(1/\varepsilon^{(d-1)/2})$ update time [7]. For the k-center problem, several approximation algorithms [14,15,16] also have been proposed.

Our results. We describe a $(2 + \sqrt{2} + \varepsilon^*)$-approximation algorithm that uses $O(d^2)$ space and $O(d)$ update time for arbitrary dimension d where ε^* is an arbitrarily small positive constant. After that we present two approximation algorithms for fixed constant dimension d. The first approximation algorithm guarantees a 3-approximation using $O(1)$ space and $O(1)$ update time, and the next one guarantees a $(1 + \varepsilon)$-approximation using $O(1/\varepsilon^d)$ space and $O(1/\varepsilon^{(d-1)/2})$ update time. One may think the last two approximation algorithms have the same complexity for $\varepsilon = 2$, but the 3-approximation algorithm only uses space polynomial in d, so it is more valuable than the $(1 + \varepsilon)$-approximation algorithm in the streaming model. Table 1 shows a summary of our results.

Dimension d	Factor	Space	Update Time
arbitrary dim. d	$(2 + \sqrt{2} + \varepsilon^*)$	$O(d^2)$	$O(d)$
constant dim. d	3	$O(1)$	$O(1)$
	$(1 + \varepsilon)$	$O(1/\varepsilon^d)$	$O^*(1/\varepsilon^{(d-1)/2})$

Table 1. Results for the smallest intersecting ball of disjoint balls problem over the single-pass streaming model. $O^*(x)$ denotes $O(x)$ amortized time, and ε^* denotes an arbitrarily small positive constant.

2 Preliminaries

Let D be a set of n pairwise interior-disjoint balls in $\mathbb{R}^d$. The balls in D arrive one by one over the single-pass stream. They are labeled in order, so b_i is a ball in D that has arrived at the i-th step, that is, $D = \{b_1, b_2, ..., b_n\}$.

Let $b(c, r)$ denote a ball centered at c of radius r, and let $c(b)$ and $r(b)$ denote the center and the radius of a ball b, respectively. We denote B^* *the optimal solution*, and c^* and r^* denote $c(B^*)$ and $r(B^*)$, respectively.

The distance between any two points p and q is denoted by $|pq|$, and the distance between any two balls b and b' is denoted by $dist(b, b') = \max\{|c(b)c(b')| - (r(b) + r(b')), 0\}$. We use $dist(p, b)$ to denote the distance between a point p and a ball b, that is, $\max\{|pc(b)| - r(b), 0\}$.

Our goal is to approximate the smallest ball B^* that intersects all the input balls.

3 $(2 + \sqrt{2} + \varepsilon^*)$-Approximation Algorithm in any dimensions d

We introduce an algorithm that guarantees $(2 + \sqrt{2} + \varepsilon^*)$-approximation factor for any d where ε^* is an arbitrarily small positive constant. It is trivial to solve the problem for $d = 1$, and the algorithm in Section 4 gives a better result for $d = 2$, so we assume that $d \geq 3$ in this section.

Lemma 1. *The radius of d concurrent interior-disjoint balls is at most $\dfrac{\sqrt{d}}{\sqrt{2(d-1)}} r$ where r is the radius of the smallest enclosing ball of the centers of the d balls.*

Proof. Let us consider d points in a ball b with radius r. To maximize the distance of the closest pair of the points, they should satisfy the following conditions.

- they should lie on the boundary of the ball,
- the distances between all the pairs should be same, and
- the hyper-plane h defined by them should contain $c(b)$.

The above imply that they are vertices of a $(d - 1)$-dimensional regular simplex on h. The side length of a $(d - 1)$-dimensional regular simplex is $2\dfrac{\sqrt{d}}{\sqrt{2(d-1)}} r$ where r is the radius of the circumscribed ball of it. Therefore the lemma holds. ⬦

We can derive the following lemma from Lemma 1. Let $c_d = \dfrac{\sqrt{2(d-1)}\sqrt{d}+d}{d-2}$.

Lemma 2. *There are at most $d - 1$ input balls such that radius of each of them is greater than $c_d r^*$.*

Proof. Assume to the contrary that there are d input balls such that radii of all of them are $c_d r^* + \varepsilon^*$ where ε^* is an arbitrarily small positive constant. Their centers should lie in $b(c^*, (1 + c_d)r^* + \varepsilon^*)$ by the problem definition. Let $r = (1 + c_d)r^* + \varepsilon^*$, then for $d \geq 3$

$$\frac{\sqrt{d}}{\sqrt{2(d-1)}} r = c_d r^* + \frac{\sqrt{d}}{\sqrt{2(d-1)}} \varepsilon^* < c_d r^* + \varepsilon^*$$
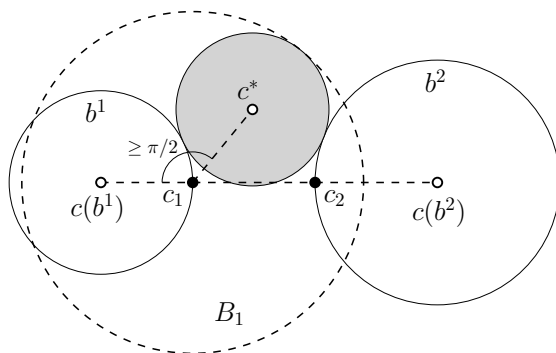
Fig. 2. The two input balls b^1 and b^2, B^* (the gray ball), and one of our solution B_1 (the dashed ball).

By Lemma 1, at least one pair of the balls should intersect each other, a contradiction. ⌑

We propose a simple approximation algorithm by using Lemma 2 as follows. We keep the first d input balls, and then find the smallest ball b_{min} among them. We set the center of our solution to $c(b_{min})$, and then expend radius of our solution whenever a new input ball b arrives that does not intersect our solution.

By Lemma 2, $r(b_{min})$ is at most $c_d r^*$, so our solution guarantees an approximation factor $2 + c_d$. Because $\sqrt{2(d-1)}\sqrt{d} < \sqrt{2}(d-1) + \sqrt{d-1}$ for any $d \geq 3$ the following equation holds.

$$c_d = \frac{\sqrt{2(d-1)}\sqrt{d} + d}{d-2} < (1 + \sqrt{2}) + \frac{\sqrt{d-1} + \sqrt{2} + 2}{d-2}$$

We can use the algorithm in Section 4 for a small constant dimension, so following theorem holds.

Theorem 1. *For streaming balls in arbitrary dimensions d, there is an algorithm that guarantees a $(3 + \sqrt{2} + \varepsilon^*)$-approximation to the smallest intersecting ball of disjoint balls problem using $O(d^2)$ space and $O(d)$ update time where ε^* is an arbitrarily small positive constant.*

3.1 Improved Approximation Algorithm

The above algorithm can be improved by maintaining two solutions B_1 and B_2 as follows. See Figure 2. We keep the first $d+1$ input balls, and then find the two smallest balls b^1 and b^2 among them. Let s be the line segment connecting $c(b^1)$ and $c(b^2)$ (remember that $c(b^1)$ and $c(b^2)$ are the centers of B_1 and B_2, respectively). We set $c(B_1)$ and $c(B_2)$ to $b^1 \cap s$ and $b^2 \cap s$, respectively, and then expand each of them to make it intersects all the input balls. Our solution B at the end is the smaller one between B_1 and B_2.
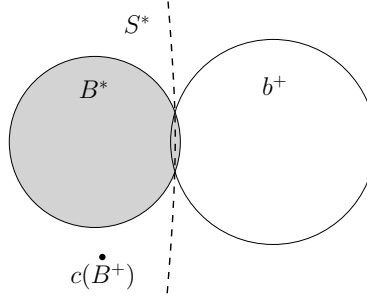
Fig. 3. Proof of Lemma 3

Let us consider the correctness and the approximation factor of the above algorithm. Obviously, our solution B intersects all the input balls. As shown in Figure 2, one of $\angle c(b^1)c_1c^*$ and $\angle c(b^2)c_2c^*$ is greater than or equal to $\pi/2$. Without loss of generality, let us assume that $\angle c(b^1)c_1c^* \geq \pi/2$. The approximation factor of our solution is $(|c_1c^*| + r^*)/r^*$, and

$$|c_1c^*|^2 \leq |c(b^1)c^*|^2 - r(b^1)^2 \leq (r^* + r(b^1))^2 - r(b^1)^2 = (r^*)^2 + 2r^*r(b^1)$$

by the Pythagorean theorem. By Lemma 2, $r(b^1) \leq (1 + \sqrt{2} + \varepsilon^*)r^*$, so

$$|c_1c^*| \leq \sqrt{1 + 2(1 + \sqrt{2})}r^* + 2\varepsilon^*r^* = (1 + \sqrt{2})r^* + 2\varepsilon^*r^*$$

which proves the following theorem.

Theorem 2. *For streaming balls in arbitrary dimensions d, there is an algorithm that guarantees a $(2 + \sqrt{2} + \varepsilon^*)$-approximation to the smallest intersecting ball of disjoint balls problem using $O(d^2)$ space and $O(d)$ update time.*

Because of the curse of dimensionality, there can be $d+1$ balls each of radius is slightly smaller than c_dr^* in high dimensions, and both of $\angle c(b^1)c_1c^*$ and $\angle c(b^2)c_2c^*$ can be $\pi/2$, so the analysis of our algorithm is tight. Next two sections introduce approximation algorithms in fixed constant dimensions d.

4   3-Approximation Algorithm in fixed dimensions d

In this section, we introduce a 3-approximation algorithm in fixed constant dimensions d. The following lemma is the heart of the algorithm.

Lemma 3. *Any set D' of $d + 2$ input balls satisfies the one of the following conditions.*

1. *there is a ball b in D' such that $r(b) \leq r^*$, or*
2. *the smallest ball B^+ that intersects all the balls in D' intersects B^*.*

Proof. If Condition 1 holds, then we are done. So let us assume that radius of each of the balls in D' is greater than r^*. B^+ is determined by at most $d+1$ balls, so there is a ball $b^+ \in D'$ that does not determine B^+.

Let us consider a bisector of b^+ and B^* (See Figure 3). The bisector subdivides the space into two parts; all the points in one of them are closer to B^*, and all the points in the other part are closet to b^+. Let S^* be the subspace defined by the bisector such that $dist(p, B^*) \leq dist(p, b^+)$ for all points p in S^*. Since $r^* < r(b^+)$, S^* is convex.

All the balls in $D' \setminus \{b^+\}$ intersect B^* and do not intersect interior of b^+. It means that $dist(c(b), B^*) \leq dist(c(b), b^+)$ for all $b \in D' \setminus \{b^+\}$, so $c(b)$ is contained in S^*. B^+ is determined by balls in $D' \setminus \{b^+\}$, so $c(B^+)$ is also contained in S^*.[1]

By the definition of S^*, $dist(c(B^+), B^*) \leq dist(c(B^+), b^+)$, and $dist(c(B^+), b^+) \leq r(B^+)$ by the definition of B^+. Then $dist(c(B^+), B^*) \leq r(B^+)$, which proves the lemma. □

Our algorithm is as follows. We keep two solutions simultaneously; one is based on the assumption that Condition 1 in Lemma 3 holds, and the other is based on the assumption that Condition 2 holds. We choose the better one among them at the end.

The solution based on Condition 1 can be computed as follows. We keep the first $d+2$ input balls, and then find the smallest ball b_{min} among them. We set the center of our solution to $c(b_{min})$, and then expend the radius of our solution whenever a new input ball arrives that does not intersect our solution while keeping the center unchanged.

The solution based on Condition 2 can be computed by the almost same way except the way to choose the center of our solution. We keep the first $d+2$ input balls, and then compute the optimal solution B^+ for them. We set the center of our solution to $c(B^+)$. The remaining parts are the same as our solution based on Condition 1.

Both of our solutions obviously intersects all the input balls. By Lemma 3 and the problem definition, one of b_{min} and B^+ has radius $r \leq r^*$ and intersects B^*, which means that the algorithm guarantees a 3-approximate solution. Our algorithm spends $O(1)$ time [3] to compute B^+ and $O(1)$ update time by using $O(1)$ space.

Theorem 3. *For streaming balls in fixed constant dimensions d, there is an algorithm that guarantees a 3-approximation to the smallest intersecting ball of disjoint balls problem using $O(1)$ space and $O(1)$ update time.*

Note that the space of our algorithm does not depend on n, and it has polynomial dependency on d.

Now we show that our approximation factor analysis for the algorithm is tight by showing an example in $\mathbb{R}^3$ as follow. See Figure 4. Let us consider the first five input balls. If they satisfy Condition 1 in Lemma 3, it is trivial to show tightness, so let us assume that they only satisfy Condition 2.

The centers of the first two input balls b_1 and b_2 are at $(0, 0, a)$ and $(0, 0, -a)$, respectively, where a is an arbitrarily large positive constant. We assume that $dist(b_1, b_2) = 2r^* - \varepsilon^*$ where ε^* is an arbitrarily small positive constant, and they determine B^+; $c(B^+) = (0, 0, 0)$. The remaining three input balls b_3, b_4 and b_5 are slightly greater

---

[1] $c(B^+)$ is contained in the convex hull of centers of balls in $D' \setminus \{b^+\}$, and the convex hull is contained in S^*.
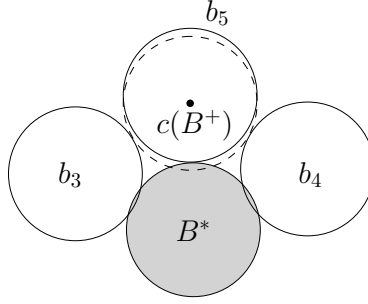
Fig. 4. A tight example of Theorem 3: three input balls b_3, b_4 and b_5, the optimal solution B^*, and the optimal solution B^+ (dashed) for the first five input balls.

than B^* and their centers are on the xy-plane. Let us assume that c^* is also on the xy-plane, so they look like Figure 4 on the xy-plane. The figure shows that the five input balls are interior-disjoint, and they do not satisfy Condition 1. As you see, we can make the intersecting area of B^+ and B^* as small as possible, which proves the tightness of the analysis.

5 $(1 + \varepsilon)$-Approximation Algorithm in fixed dimensions d

We present a $(1+\varepsilon)$-approximation algorithm in fixed constant dimensions d in this section. Before we propose our algorithm, we describe a $(1 + \varepsilon)$-approximation algorithm for the smallest enclosing ball of points problem.

Definition 1. *[17] Given a double-argument measure $w(P, x) = \max_{p,q \in P} (p - q) \cdot x$ that is monotone in its first argument, a subset $R \subset P$ is called an ε-core-set of P over all vectors $x \in \mathbb{R}^d$ if $w(R, x) \geq (1 - \varepsilon)w(P, x)$ for all x.*

For streaming points in any fixed dimensions d, one can devise an ε-core-set by maintaining extreme points along a number of different directions using $O(1/\varepsilon^{(d-1)/2})$ space and $O(1/\varepsilon^{(d-1)/2})$ update time [17]. A $(1 + \varepsilon)$-approximate solution can be obtained by increasing the radius of the optimal solution for points in an ε-core-set by $2\varepsilon r_p^*$ where r_p^* is the optimal radius of the smallest enclosing ball of points problem.

 The basic idea of our algorithm is as follows. Let $D_<$ be a subset of D such that $r(b) \leq \varepsilon r^*$ for $b \in D_<$, and $C_< = \{c(b) \mid b \in D_<\}$. We also denote by $\varepsilon C_<$ an ε-core-set of $C_<$, and $\varepsilon D_< = \{b \mid c(b) \in \varepsilon C_<\}$. We maintain $\varepsilon D_<$ and all balls in $D \setminus D_<$. For $\varepsilon D_<$, the following lemma hold.

Lemma 4. *Let B be the optimal solution for balls in $\varepsilon D_<$, then $r(B) \leq r^*$ and $dist(B, b) \leq 5\varepsilon r^*$ for any $b \in D$ if $0 < \varepsilon < 1$.*

Proof. The radius of B is smaller than or equal to r^* since $\varepsilon D_< \subset D$.

 Let $b \in D$ be a ball that does not intersect B. It means that $b \notin \varepsilon D_<$. By Definition 1, there is a ball $b' \in \varepsilon D_<$ such that $(c(b) - c(b')) \cdot x \leq \varepsilon \cdot w(C_<, x)$ where

$x = \frac{(c(b)-c(B))}{|c(b)c(B)|}$. Let p be a point in $B \cap b'$. The following equation proves the lemma.

$$dist(B,b) \leq dist(B,c(b)) \leq (c(b)-p) \cdot x = (c(b)-c(b')) \cdot x + (c(b')-p) \cdot x$$
$$\leq \varepsilon \cdot w(C_<,x) + \varepsilon r^* \leq \varepsilon(2r^* + 2\varepsilon r^*) + \varepsilon r^* \leq 5\varepsilon r^*$$

□

So, we can get a $(1 + 5\varepsilon)$-approximate solution by increasing the radius of B by $5\varepsilon r^*$ if $0 < \varepsilon < 1$. We present a 3-approximation algorithm in Section 4, so $0 < 5\varepsilon < 2$ is enough.

Now, let us consider our algorithm in detail. We partition D into $O(\varepsilon^d)$ subsets. Let D_i be the subset that contains $\lceil 1/\varepsilon^d \rceil$ input balls in order from the $((i-1)\cdot\lceil 1/\varepsilon^d \rceil +1)$-th input ball. We process D_i one by one in order.

For D_1, we compute the optimal solution B for balls in it. We find balls in D_1 each ball b of them satisfies $r(b) \leq \varepsilon r(B)$, and then compute $\varepsilon D_<$ by considering their centers. We insert all the other balls in a set $D_>$. We only maintain $\varepsilon D_<$ and $D_>$ for the next step.

For D_2, we compute the optimal solution B for the balls in $D_2 \cup D_> \cup \varepsilon D_<$. Similarly, we find balls each of radius smaller than or equal to $\varepsilon r(B)$ from $D_2 \cup D_>$, and then update $\varepsilon D_<$ by considering them. We delete such balls from D_2 and $D_>$, and then update $D_>$ by inserting all remaining balls in D_2. We repeat this process for all D_i where i is an integer between 1 and $O(\varepsilon^d)$.

At the end of the algorithm, we compute the optimal solution B for balls in $D_> \cup \varepsilon D_<$, and then increase the radius of B by $5\varepsilon r^+$ where r^+ is the radius from the algorithm in Section 4. We use the algorithm in Section 4 simultaneously to compute r^+. Finally, we report B as our solution.

The correctness of our algorithm immediately follows from Lemma 4. In each step, a ball b that satisfies $r(b) \leq \varepsilon r(B)$ also guarantees that $r(b) \leq \varepsilon r^*$. At the end of the algorithm, we maintain $\varepsilon D_<$ and all the other balls, so the optimal solution for them guarantees that $r(B) \leq r^*$ and $dist(B,b) \leq 5\varepsilon r^*$ by Lemma 4. The radius r^+ guarantees that $r^* \leq r^+ \leq 3r^*$ by Theorem 3, so $5\varepsilon r^* \leq 5\varepsilon r^+ \leq 15\varepsilon r^*$. Therefore after increasing the radius of B by $5\varepsilon r^+$, B intersects all the input balls. The approximation factor is $(1 + 15\varepsilon)$, and we can get a $(1 + \varepsilon')$ approximation algorithm by adjusting a parameter $\varepsilon' = 15\varepsilon$.

Let us analyze the complexity of the algorithm. To maintain $\varepsilon D_<$, the algorithm uses $O(1/\varepsilon^{(d-1)/2})$ space and $O(1/\varepsilon^{(d-1)/2})$ update time [7]. The size of $D_>$ can be computed by the following lemma.

Lemma 5. *For given a ball b of radius r, there are at most $O(1/\varepsilon^d)$ interior-disjoint balls that intersect b if radius of each of them is greater than or equal to εr in fixed constant dimensions d.*

Proof. We are going to prove the lemma for the interior-disjoint balls each of radius εr. Obviously, if there are at most $O(1/\varepsilon^d)$ such balls, then the lemma holds. Let us consider a ball $b' = b(c(b), r + 2\varepsilon r)$. A ball that intersects b should be contained in b'. We can compute the maximum number of the interior-disjoint balls that are contained

in b' by considering their volumes. The volume of b' is $\Theta((r + 2\varepsilon r)^d)$, and the volume of a ball of radius εr is $\Theta((\varepsilon r)^d)$. The sum of the volumes of all the balls contained in b' can not exceed the volume of b', so the maximum number of the interior-disjoint balls in b' is $O((r + 2\varepsilon r)/\varepsilon r)^d) = O(1/\varepsilon^d)$, which prove the lemma. $\qquad\square$

By Lemma 5, the size of $D_>$ is $O(1/\varepsilon^d)$. In each step the algorithm holds $O(1/\varepsilon^d)$ balls. We compute the optimal solution for them in each step and it takes linear time [18], so we spend $O(1)$ amortized time per update.

Theorem 4. *For streaming balls in fixed constant dimensions d, there is an algorithm that guarantees a $(1 + \varepsilon)$ approximation to the smallest intersecting ball of disjoint balls problem using $O(1/\varepsilon^d)$ space and $O(1/\varepsilon^{(d-1)/2})$ amortized update time.*

6 Conclusion

In this paper, we introduced three approximation algorithms for the smallest intersecting ball of disjoint balls problem. One of them is for the problem in any arbitrarily dimensions, and the others are for the problem in fixed constant dimensions. As the exact problem is very difficult (no polynomial algorithm is known if d is not constant), approximation seems to be the only way forward.

We do not know any better lower bound for the worst-case approximation ratio than $(1 + \sqrt{2})/2 > 1.207$ that is a lower bound for the smallest enclosing ball of points problem in the streaming model if we use space only polynomially bounded in d [6]. We believe that this lower bound is not tight for our problem and it can be improved.

A natural extension of our algorithm is to allow the input balls to overlap. However, this poses a great number of challenges since the size of the optimal answer could be zero (when a point pierces all the balls). But if we allow overlapping with some restrictions it may be possible to solve the problem. Another interesting question is how to improve our results in static setting. Even in the static setting, no approximation algorithms are known for the problem except our results in this paper.

References

1. Mordukhovich, B., Nam, N., Villalobos, C.: The smallest enclosing ball problem and the smallest intersecting ball problem: existence and uniqueness of solutions. Optimization Letters **7**(5) (2013) 839–853
2. Löffler, M., van Kreveld, M.: Largest bounding box, smallest diameter, and related problems on imprecise points. Computational Geometry **43**(4) (2010) 419 – 433
3. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica **16**(4-5) (1996) 498–516
4. Ahn, H.K., Kim, S.S., Knauer, C., Schlipf, L., Shin, C.S., Vigneron, A.: Covering and piercing disks with two centers. Computational Geometry **46**(3) (2013) 253 – 262
5. Zarrabi-Zadeh, H., Chan, T.: A simple streaming algorithm for minimum enclosing balls. In: Proc. of the 18th Canadian Conference on Computational Geometry. (2006) 139–142
6. Agarwal, P.K., Sharathkumar, R.: Streaming algorithms for extent problems in high dimensions. In: Proc. of the 21st ACM-SIAM Symposium on Discrete Algorithms. SODA '10 (2010) 1481–1489

7. Chan, T.M., Pathak, V.: Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. Computational Geometry **47**(2, Part B) (2014) 240 – 247

8. Bâdoiu, M., Clarkson, K.L.: Smaller core-sets for balls. In: Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms. SODA '03 (2003) 801–802

9. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman (1979)

10. Chan, T.: More planar two-center algorithms. Computational Geometry **13**(3) (1999) 189–198

11. Agarwal, P., Avraham, R., Sharir, M.: The 2-center problem in three dimensions. In: Proc. of the 26th ACM Sympos. Computational Geometry. (2010) 87–96

12. Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38** (1985) 293–306

13. Feder, D., Greene, D.: Optimal algorithms for approximate clustering. In: Proc. of the 20th ACM Symposium on Theory of Computing. (1988) 434–444

14. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. SIAM Journal on Computing **33**(6) (2004) 1417–1440

15. Guha, S.: Tight results for clustering and summarizing data streams. In: Proc. of the 12th International Conference on Database Theory. (2009) 268–275

16. McCutchen, R., Khuller, S.: Streaming algorithms for k-center clustering with outliers and with anonymity. In: Proc. of the 11th International Workshop, APPROX 2008 and the 12th International Workshop, RANDOM 2008. (2008) 165–178

17. Chan, T.M.: Faster core-set constructions and data-stream algorithms in fixed dimensions. Computational Geometry **35**(12) (2006) 20 – 35

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