Streaming and Dynamic Algorithms for Minimum Enclosing Balls in High Dimensions

Timothy M. Chan and Vinayak Pathak

School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada,
{tmchan,vpathak}@uwaterloo.ca

Abstract. At SODA'10, Agarwal and Sharathkumar presented a streaming algorithm for approximating the minimum enclosing ball of a set of points in d-dimensional Euclidean space. Their algorithm requires one pass, uses $O(d)$ space, and was shown to have approximation factor at most $(1+\sqrt{3})/2+\varepsilon \approx 1.3661$. We prove that the same algorithm has approximation factor less than 1.22, which brings us much closer to a $(1+\sqrt{2})/2 \approx 1.207$ lower bound given by Agarwal and Sharathkumar. We also apply this technique to the dynamic version of the minimum enclosing ball problem (in the non-streaming setting). We give an $O(dn)$-space data structure that can maintain a 1.22-approximate minimum enclosing ball in $O(d \log n)$ expected amortized time per insertion/deletion.

1 Introduction

In this paper, we study a fundamental and well-known problem in computational geometry: Given a set P of points in $\mathbb{R}^d$, find the ball with the smallest radius that contains all points in P. This is known as the *minimum enclosing ball* or the *1-center* problem and has various applications, for example, in clustering and facility location. We will not survey the many known exact algorithms for the problem, as the focus of the paper is on *approximation* algorithms in the streaming and the dynamic setting.

In the standard *streaming model*, we consider algorithms that are allowed one pass over the input and can store only a small (usually polylogarithmic) amount of information at any time, as points arrive one at a time. In low constant dimensions, it is not difficult to devise a streaming algorithm that computes a $(1+\varepsilon)$-approximation to the minimum enclosing ball using $O(1/\varepsilon^{(d-1)/2})$ space, by maintaining extreme points along a number of different directions. In fact, streaming techniques for *ε-kernels* [1, 8, 3, 11] allow many other similar geometric optimization problems to be solved with approximation factor $1+\varepsilon$ using $1/\varepsilon^{O(d)}$ space. However, these techniques do not work well in high dimensions because of the exponential dependencies on d.

In high dimensions, there is a trivial streaming algorithm with approximation factor 2: fix the center of the ball B at an arbitrary input point p_0 (say the first point), and whenever a new point p arrives that lies outside B, expand B to include p while keeping the center unchanged (see Section 2.1). Zarrabi-Zadeh

and Chan [12] gave a nontrivial analysis showing that another equally simple streaming algorithm achieves approximation factor 3/2: whenever a new point p arrives that lies outside the current ball B, replace B by the smallest ball enclosing $B \cup \{p\}$. An advantage of these simple algorithms is that they avoid the exponential dependencies on d, using asymptotically the minimal amount of storage, namely, $O(d)$ space. (We assume that a unit of space can hold one coordinate value.)

Most recently, Agarwal and Sharathkumar [2] described a new streaming algorithm that also required just $O(d)$ space but with an even better approximation factor. They proved that the factor is upper-bounded by $(1+\sqrt{3})/2+\varepsilon \approx 1.3661$, where as usual, ε denotes an arbitrarily small positive constant. They also proved a complementary lower-bound result: any algorithm in the one-pass streaming model with space polynomially bounded in d has worst-case approximation factor at least $(1+\sqrt{2})/2 > 1.207$. The gap between 1.3661 and 1.207 raises an interesting question of what the best constant could be. It also reveals our current lack of general understanding on high-dimensional geometric problems in the streaming model, as the minimum enclosing ball problem is one of the most basic and simplest to consider.

In this paper, we describe an improved upper bound of 1.22 for minimum enclosing ball in the streaming model. The improvement is actually achieved by the same algorithm as Agarwal and Sharathkumar's; our contribution lies in a better analysis of their algorithm. As intermediate solutions, we first present two simple proofs, one yielding upper bounds of $4/3 + \varepsilon \approx 1.333\ldots$[1] and another yielding $16/13 + \varepsilon \approx 1.2307$. The 1.22 bound is obtained through more involved numerical calculations done with the assistance of a computer program.

In the second part of the paper, we investigate the *dynamic* version of the approximate minimum enclosing ball problem. Here, we are interested in data structures that support insertions and deletions of input points efficiently. Unlike in the streaming model, linear space is acceptable. As before, the problem is not difficult in low dimensions: one can maintain a $(1+\varepsilon)$-approximation to the minimum enclosing ball in $O(1/\varepsilon^{(d-1)/2} \log n)$ time with a data structure using $O(n/\varepsilon^{(d-1)/2})$ space, by keeping track of extreme points along various directions. The $\log n$ factor in the update time can be reduced to constant in the word RAM model [9].

In the high-dimensional case, it is possible to dynamize the trivial factor-2 method by using a simple randomization trick (see Section 3.1), but we are not aware of any prior work on efficient dynamic data structures that achieve approximation factor smaller than 2 and avoid exponential dependency on d.

We show that Agarwal and Sharathkumar's approach, which was originally intended for streaming, can be applied to the dynamic problem as well, if combined in the right way with ideas from other known techniques: specifically,

[1] Independent to our work, Agarwal and Sharathkumar (personal communication, Dec. 2010) have also found a proof of the 4/3 upper bound, which will appear in the journal version of their paper. Even compared against the 4/3 bound, our 1.22 bound is a substantial improvement.

Bădoiu and Clarkson's static method for high-dimensional minimum enclosing ball [6], and Chan's dynamization strategy for low-dimensional ε-kernels [9]. The resulting data structure requires $O(dn)$ space and supports updates in $O(d \log n)$ expected amortized time. Our analysis of Agarwal and Sharathkumar's technique implies that the same 1.22 upper bound carries over to this dynamic data structure.

2 Streaming MEB

2.1 Preliminaries and Agarwal and Sharathkumar's algorithm

Let P be a set of points in $\mathbb{R}^d$. We use $\mathrm{MEB}(P)$ to denote the minimum enclosing ball of the set P. For a ball B, we use $r(B)$ and $c(B)$ to denote its radius and center respectively. αB stands for the ball with center at $c(B)$ and radius equal to $\alpha r(B)$.

A very simple factor-2 streaming algorithm for approximating the MEB works as follows. Let the first point be p_0. Find the point p_1 in P that is farthest away from p_0. This can be implemented by a one-pass streaming algorithm. Return the ball centered at p_0 of radius $\|p_0p_1\|$. This ball clearly encloses P. The approximation factor is at most 2, since the MEB of P must enclose p_0 and p_1, and any ball that encloses p and q must have radius at least $\|p_0p_1\|/2$.

If more than one pass is allowed, we can get better ratios. In particular, Bădoiu and Clarkson [6] (building on prior work by Bădoiu, Har-Peled, and Indyk [7]) proved that we can achieve an approximation factor of $1+\varepsilon$ in $O(1/\varepsilon)$ passes. The algorithm works as follows. Pick a point $p_0 \in P$. Next, pick p_1 to be the point farthest from p_0 in P. In general, pick p_j to be the point farthest from $c(\mathrm{MEB}(\{p_0, \ldots, p_{j-1}\}))$ in P. It was shown that after $\lceil 2/\varepsilon \rceil$ iterations, the set K of $O(1/\varepsilon)$ chosen points satisfies the following *coreset* property, which implies that $r(\mathrm{MEB}(K))$ (computable by brute force) is a $(1+\varepsilon)$-approximation:

Definition 1. *Given a set P of points in $\mathbb{R}^d$, an* ε-coreset *of P is a subset $K \subseteq P$ such that $P \subseteq (1+\varepsilon)\mathrm{MEB}(K)$.*

Using Bădoiu and Clarkson's algorithm as a subroutine, Agarwal and Sharathkumar [2] gave a streaming algorithm for finding a $((1+\sqrt{3})/2+\varepsilon)$-factor MEB of a given set of points. The algorithm works as follows. Let the first point in the input stream be its own coreset and call the coreset K_1. Next, as long as the new arriving points lie inside $(1+\varepsilon)\mathrm{MEB}(K_1)$, do nothing. Otherwise, if p_i denotes the new point, call Bădoiu and Clarkson's algorithm on the set $K_1 \cup \{p_i\}$. This gives a new coreset K_2. In general, maintain a sequence of coresets $\mathcal{K} = \langle K_1, \ldots, K_u \rangle$ and whenever a new point p_i arrives such that it does not lie in $(1+\varepsilon)\mathrm{MEB}(K_j)$ for any j, call Bădoiu and Clarkson's algorithm on the set $\bigcup_{j=1}^{u} K_j \cup \{p_i\}$. However, doing this might make the sequence $\mathcal{K}$ too large. To reduce space, whenever a new call to the subroutine is made, the algorithm also removes some of the previous K_i's when $r(\mathrm{MEB}(K_i))$ is smaller than $O(\varepsilon)r(\mathrm{MEB}(K_u))$. Agarwal and Sharathkumar proved that this removal process

does not hurt the effectiveness of the data structure, and the following invariants are maintained, where $B_i = \text{MEB}(K_i)$:

(P1) For all i, $r(B_{i+1}) \geq (1 + \Omega(\varepsilon^2))r(B_i)$.
(P2) For all $i < j$, $K_i \subset (1 + \varepsilon)B_j$.
(P3) $P \subset \bigcup_{i=1}^{u}(1 + \varepsilon)B_i$.

The sequence $\mathcal{K}$ of coresets was called an *ε-blurred ball cover* in the paper. Property (P1) ensures that the number of coresets maintained at any time is $u = O(\log(1/\varepsilon))$. Since each coreset has size $O(1/\varepsilon)$, the total space is $O(d)$ for constant ε. Let $B = \text{MEB}(\bigcup_{i=1}^{u} B_i)$ (computable by brute force). Property (P3) ensures that $(1+\varepsilon)B$ encloses P. Using property (P2), Agarwal and Sharathkumar proved that $r(B) \leq (\frac{1+\sqrt{3}}{2} + \varepsilon)\, r(\text{MEB}(P))$, thus giving a factor-1.366 algorithm for MEB in the streaming model. We show that in fact, the approximation factor is less than 1.22. The proof amounts to establishing the following (purely geometric) theorem:

Theorem 1. *Let $K_1, \ldots, K_u$ be subsets of a point set P in $\mathbb{R}^d$, with $B_i = \text{MEB}(K_i)$, such that $r(B_i)$ is increasing over i and property* (P2) *is satisfied for a sufficiently small $\varepsilon > 0$. Let $B = \text{MEB}(\bigcup_{i=1}^{u} B_i)$. Then $r(B) < 1.22\, r(\text{MEB}(P))$.*

2.2 An improved analysis

We will prove Theorem 1 in the next few subsections. First we need the following well-known fact, often used in the analysis of high-dimesional MEB algorithms:

Lemma 1 (the "hemisphere property"). *Let P be a set of points in $\mathbb{R}^d$. There is no hemisphere of $\text{MEB}(P)$ that does not contain a point from P. In other words, assuming the origin to be at the center of $\text{MEB}(P)$, for any unit vector v, there exists a point $p \in P$ such that p lies on the boundary of $\text{MEB}(P)$ and $v \cdot p \leq 0$.*

We introduce a few notations. Without loss of generality, let $r(B) = 1$ and $c(B)$ be the origin. Let u_i be the unit vector in the direction of the center of B_i and $\sigma_{ij} = u_i \cdot u_j$ be the inner product between the vectors u_i and u_j. Let us also write $r(B_i)$ simply as r_i and set $t_i = 1/(1 - r_i)$. Note that the $t_i \geq 1$ are increasing over i.

Lemma 2. *For all $i < j$ with $t_i \leq t_j < 10$ such that B_i and B_j touch ∂B,*

$$\sigma_{ij} \;\geq\; \frac{t_j}{t_i} - t_j + t_i - O(\varepsilon).$$

Proof. Let c, c_i, c_j be the centers of the balls B, B_i, B_j respectively. Figure 2.2 shows the projection of $B, (1+\varepsilon)B_i, B_j$ onto the plane formed by c, c_i, c_j. Let p be one of the points where $(1 + \varepsilon)B_j$ intersects B_i in this plane. Applying the cosine law to the triangle $c_i c_j c$, we get

$$\|c_i c_j\|^2 = \|cc_j\|^2 + \|cc_i\|^2 - 2\|cc_i\|\|cc_j\|\sigma_{ij}. \tag{1}$$

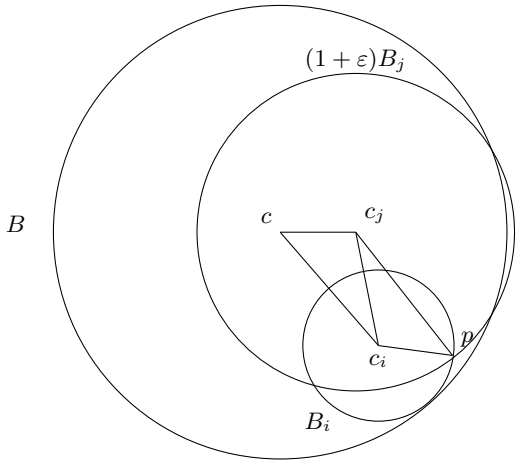

Fig. 1. Proof of Lemma 2

Next, we apply the hemisphere property to the ball $B_i = \mathrm{MEB}(K_i)$. Choosing v to be the vector $c_j - c_i$, we deduce the existence of a point $q \in K_i$ such that q lies on ∂B_i and $\angle c_j c_i q \geq \pi/2$. By property (P2) of the blurred ball cover, we know that $q \in K_i \subset (1+\varepsilon)B_j$. Since $\|c_i p\| = \|c_i q\|$ and $\|c_j p\| \geq \|c_j q\|$, we have $\angle c_j c_i p \geq \angle c_j c_i q \geq \pi/2$. This means

$$\|c_j p\|^2 \geq \|c_i c_j\|^2 + \|c_i p\|^2. \tag{2}$$

Substituting $\|c_j p\| = (1+\varepsilon)r_j, \|c_i p\| = r_i, \|cc_j\| = 1 - r_j, \|cc_i\| = 1 - r_i$ into (1) and (2) and combining them, we get

$$(1+\varepsilon)^2 r_j^2 \;\geq\; (1-r_j)^2 + (1-r_i)^2 - 2(1-r_i)(1-r_j)\sigma_{ij} + r_i^2.$$

Letting $s_i = 1 - r_i$ and $s_j = 1 - r_j$ and $t_i = 1/s_i$ and $t_j = 1/s_j$, we get

$$\begin{array}{rrcl} & (1+\varepsilon)^2(1 - 2s_j + s_j^2) & \geq & s_i^2 + s_j^2 - 2s_i s_j \sigma_{ij} + (1 - 2s_i + s_i^2) \\ \Longrightarrow & 2s_i s_j \sigma_{ij} & \geq & 2s_i^2 - 2s_i + 2s_j - O(\varepsilon) \\ \Longrightarrow & \sigma_{ij} & \geq & t_i - t_j + t_j/t_i - O(\varepsilon t_i t_j). \end{array}$$

(The assumption $t_i \leq t_j < 10$ allows us to rewrite $O(\varepsilon t_i t_j)$ as $O(\varepsilon)$.) □

2.3 Proof of factor 4/3

As a warm-up, in this subsection, we give a short proof of a weaker 4/3 upper bound on the constant in Theorem 1.

Let B_i be the largest ball that touches ∂B. Since B is the minimum enclosing ball of $\bigcup_{\ell=1}^{u} B_\ell$, by applying the hemisphere property to B with $v = u_i$ there must exist another ball B_j such that $\sigma_{ij} \leq 0$. Combining with Lemma 2, we get

$$\frac{t_i}{t_j} - t_i + t_j \;\leq\; O(\varepsilon) \quad\Longrightarrow\quad t_i \;\geq\; \frac{t_j - O(\varepsilon)}{1 - 1/t_j}.$$

Since $t_j \geq 1$, the minimum value achievable by t_i that satisfies the above inequality can be easily found to be $4 - O(\varepsilon)$ (attained when $t_j \approx 2$). This translates to a minimum value of $3/4 - O(\varepsilon)$ for $r_i = 1 - 1/t_i$. Since $r(\text{MEB}(P)) \geq r_i$ and $r(B) = 1$, this proves a version of Theorem 1 with the constant $4/3 + O(\varepsilon)$.

Remark: We have implicitly assumed that $t_j \leq t_i < 10$ when applying Lemma 2, but this is without loss of generality since $t_i \geq 10$ would imply $r_i > 0.99$ giving an approximation factor of ≈ 1.01.

2.4 Proof of factor 16/13

In attempting to find an example where the 4/3 bound might be tight, one could set $t_i = 4$ and $t_j = 2$, which implies $\sigma_{ij} \approx 0$ by Lemma 2, i.e., u_i and u_j are nearly orthogonal. However, by the hemisphere property, B would not be defined by the 2 balls B_i, B_j alone. This suggests that an improved bound may be possible by considering 3 balls instead of just 2, as we will demonstrate next.

Let B_i be the largest ball that touches ∂B, and B_j be the smallest ball that touches ∂B. Let $\alpha \geq 0$ be a parameter to be set later. By applying the hemisphere property to B with $v = u_i + \alpha u_j$, there must exist a k such that B_k touches ∂B and $u_k \cdot (u_i + \alpha u_j) \leq 0$. This means

$$\sigma_{ik} + \alpha \sigma_{jk} \leq 0. \tag{3}$$

Note that $t_j \leq t_k \leq t_i$. By Lemma 2, we get

$$\begin{aligned} &\frac{t_i}{t_k} - t_i + t_k + \alpha \left(\frac{t_k}{t_j} - t_k + t_j \right) \;\leq\; O(\varepsilon) \\ &\implies t_i \;\geq\; \frac{t_k + \alpha(t_k/t_j - t_k + t_j) - O(\varepsilon)}{1 - 1/t_k} \;\geq\; \frac{t_k + \alpha(2\sqrt{t_k} - t_k) - O(\varepsilon)}{1 - 1/t_k}. \end{aligned}$$

The last step follows since the minimum of $t_k/x + x$ is achieved when $x = \sqrt{t_k}$ (e.g., by the A.M.–G.M. inequality). The final expression from the last step is in one variable, and can be minimized using standard techniques. Obviously, the minimum value depends on α. As it turns out, the best bound is achieved when $\alpha = 4/3$ and the minimum value is $16/3 - O(\varepsilon)$ (attained when $t_k \approx 4$). Thus, $t_i \geq 16/3 - O(\varepsilon)$, implying $r_i = 1 - 1/t_i \geq 13/16 - O(\varepsilon)$ and an upper bound of $16/13 + O(\varepsilon)$ in Theorem 1.

2.5 Proof of factor 1.22

For our final proof of Theorem 1, the essential idea is to consider 4 balls instead of 3.

As before, let B_i be the largest ball that touches ∂B, and B_j be the smallest ball that touches ∂B. Choose a parameter $\alpha = \alpha(t_j) \geq 0$; unlike in the previous subsection, we find that making α dependent on t_j can help. By the hemisphere property, there must exist a B_k that touches ∂B while satisfying (3): $\sigma_{ik} + \alpha\sigma_{jk} \leq$

0. By applying the hemisphere property once more with $v = \beta u_i + \gamma u_j + u_k$, for every $\beta, \gamma \geq 0$, there must exists a B_ℓ that touches ∂B satisfying

$$\beta\sigma_{i\ell} + \gamma\sigma_{j\ell} + \sigma_{k\ell} \;\leq\; 0. \tag{4}$$

We prove that with Lemma 2, these constraints force $t_i > 5.54546$, implying $r_i = 1 - 1/t_i > 0.8197$ and the claimed 1.22 bound in Theorem 1. We need a noticeably more intricate argument now, to cope with this more complicated system of inequalities. Assume $t_i \leq \tau := 5.54546$.

Note that 2 cases are possible: $t_j \leq t_k \leq t_\ell \leq t_i$ or $t_j \leq t_\ell \leq t_k \leq t_i$. We first eliminate the variable ℓ in (4). By (4), we have $\forall \beta, \gamma \geq 0$:

$$\left[\exists t_\ell \in [t_k, \tau] : \beta\left(\frac{\tau}{t_\ell} - \tau + t_\ell\right) + \gamma\left(\frac{t_\ell}{t_j} - t_\ell + t_j\right) + \frac{t_\ell}{t_k} - t_\ell + t_k \leq O(\varepsilon)\right] \vee$$
$$\left[\exists t_\ell \in [t_j, t_k] : \beta\left(\frac{\tau}{t_\ell} - \tau + t_\ell\right) + \gamma\left(\frac{t_\ell}{t_j} - t_\ell + t_j\right) + \frac{t_k}{t_\ell} - t_k + t_\ell \leq O(\varepsilon)\right].$$

Observe that in each of the 2 cases, multiplying the left hand side by t_ℓ yields a quadratic inequality in t_ℓ of the form $at_\ell^2 + bt_\ell + c \leq 0$. (The $O(\varepsilon)$ terms are negligible.) In the first case, $a = \beta + \gamma/t_j - \gamma + 1/t_k - 1$, $b = -\beta\tau + \gamma t_j + t_k$, and $c = \beta\tau$; in the second case, $a = \beta + \gamma/t_j - \gamma + 1$, $b = -\beta\tau + \gamma t_j - t_k$, and $c = \beta\tau + t_k$. The variable t_ℓ can then be eliminated by the following rule: $(\exists x \in [x_1, x_2] : ax^2 + bx + c \leq 0)$ iff $(ax_1^2 + bx_1 + c \leq 0) \vee (ax_2^2 + bx_2 + c \leq 0) \vee [(a \geq 0) \wedge (b^2 \geq 4ac) \wedge (2ax_1 \leq -b \leq 2ax_2)]$.

For β, we try two fine-tuned choices: (i) $\beta = -\gamma(\tau/t_j - \tau + t_j) - (\tau/t_k - \tau + t_k) + O(\varepsilon)$ (which is designed to make the above inequality tight at $t_\ell = \tau$), and (ii) a root β of the equation $b^2 = 4ac$ where a, b, c are the coefficients of the first quadratic inequality in the preceding paragraph (for fixed t_j, t_k, γ, this is a quadratic equation in β). As it turns out, these two choices are sufficient to derive the contradiction at the end.

Three variables γ, t_j, t_k still remain and the function $\alpha(t_j)$ has yet to be specified. At this point, it is best to switch to a numerical approach. We wrote a short C program to perform the needed calculations. For γ, we try a finite number of choices, from 0 to 1 in increments of 0.05, which are sufficient to derive the desired contradiction. For (t_j, t_k), we divide the two-dimensional search space into grid cells of side length 0.0005. For each grid cell that intersects $\{t_k \leq t_j\}$, we lower-bound the coefficients of the above quadratic inequalities over all (t_j, t_k) inside the cell, and attempt to obtain a contradiction with (4) by the strategy discussed above. If we are not able to get a contradiction for the cell this way, we turn to (3), which implies

$$\frac{\tau}{t_k} - \tau + t_k + \alpha\left(\frac{t_k}{t_j} - t_k + t_j\right) \;\leq\; O(\varepsilon);$$

from this inequality, we can generate an interval of α values that guarantees a contradiction in the (t_j, t_k) cell. We set $\alpha(t_j)$ to any value in the intersection

of all α-intervals generated in the grid column of t_j. After checking that the intersection is nonempty for each grid column, the proof is complete.

Remarks: Our analysis of the system of inequalities derived from (3) and (4) is close to tight, as an example shows that these inequalities cannot yield a constant better than 1.219 regardless of the choice of the function $\alpha(t_j)$: Consider $t_i = 5.56621$ and $t_j = 2$. If $\alpha < 1.15$, pick $t_k = 2.67$; otherwise, $t_k = 5.08$. In either case, by solving a 2-variable, 100-constraint linear program in β and γ, one can verify that $\forall \beta, \gamma \geq 0$, there exists a t_ℓ from a discrete set of 100 uniformly spaced values in $[t_k, t_i]$ and $[t_j, t_k]$ such that the inequality derived from (4) is satisfied.

By choosing B_k and B_ℓ more carefully, one could add in the constraints $\sigma_{ij} \leq 0$, $\sigma_{ij} \leq \sigma_{ik}$, $\sigma_{ij} \leq \sigma_{i\ell}$, and $\sigma_{ik} + \alpha\sigma_{jk} \leq \sigma_{i\ell} + \alpha\sigma_{j\ell}$, though $t_j \leq t_k, t_\ell$ is no longer guaranteed; however, the system of inequalities becomes even harder to optimize, and we suspect that any improvements would be very small. Likewise, an analysis involving 5 or more balls does not seem to be worth the effort, until new ideas are found to simplify matters.

3 Dynamic MEB

3.1 Preliminaries and a dynamic coreset technique

In this section, we investigate how to maintain the MEB of points in high dimensions if both insertions and deletions are allowed.

The simple factor-2 streaming algorithm from Section 2 can be modified to give a factor-2 dynamic algorithm as follows. In the preprocessing stage, pick any random point p_0 from the point set P uniformly and arrange the rest of the points in a priority queue with the key being the distance of the point from p_0. Let's call p_0 the "anchor point." To insert a new point, simply insert it into the priority queue. This takes time $O(\log n)$, where n is the number of points. The MEB returned at any time is the ball centered at p_0 and having a radius equal to the maximum key. To delete a point, remove it from the priority queue if the point being deleted is not the anchor point itself. Otherwise, rebuild the whole data structure by picking a new random anchor point p and arranging the rest in a priority queue. Since the choice of the anchor point is random, the probability with which it will be deleted is $1/n$. Therefore the expected cost of deletion is $\frac{1}{n}O(n \log n) + O(\log n) = O(\log n)$. The space used is linear. (The update time can be reduced to $O(1)$ in the word RAM model by using an approximate priority queue [9].)

To obtain a ratio better than 2, we modify a dynamization technique by Chan [9]. His method was originally for maintaining a different type of coresets (called *ε-kernels* [1]) which can be applied to many problems in low dimensions, such as computing minimum-volume (non-axis-aligned) bounding boxes. We outline his method here and point out the difficulty in adapting it for high-dimensional MEB.

The starting point is a simple constant-factor approximation algorithm for the minimum bounding box [1, 4]. Pick a point $p_0 \in P$. This is the first anchor point. Next, let p_1 be the point farthest from p_0 in P. In general, pick point p_j to be the point farthest from $\mathrm{aff}\{p_0, \ldots, p_{j-1}\}$, where $\mathrm{aff}\, S$ denotes the affine hull of a set S. The resulting anchor points $p_0, \ldots, p_d$ form a coreset whose minimum bounding box approximates the minimum bounding box of P to within $O(1)$ factor. The factor can be reduced to $1 + \varepsilon$ by building a grid along a coordinate system determined by the anchor points; the size of the coreset increases to $O(\varepsilon^{-d})$.

Now, to make this algorithm dynamic, the approach is to choose the anchor points in some random way and then whenever an anchor point is deleted, rebuild the whole data structure. Because of the randomness, the deleted point will be an anchor point with only a low probability. Thus instead of choosing p_j to be the point farthest from $\mathrm{aff}\{p_0, \ldots, p_{j-1}\}$, we pick p_j uniformly at random from the set A_j of $\alpha|P|$ farthest points from $\mathrm{aff}\{p_0, \ldots, p_{j-1}\}$ and discard A_j. Thus, after picking all the anchor points, we obtain a set $R = \bigcup_j A_j$ of all discarded points. Since R is not "served" by the anchor points chosen, we recurse on R. Since $|R|$ is a fraction less than $|P|$ if the constant α is sufficiently small, this gives us a collection of $O(\log n)$ coresets. The final coreset returned is the union of all of them. Insertions can be incorporated in a standard way, analogous to the *logarithmic method* [5].

The above technique cannot be directly applied to high-dimensional MEB because of the exponential dependency of the grid size on d. Also, with our weaker form of coreset from Definition 1 for MEB, the union of coresets of a collection of subsets is not necessarily a good coreset for the whole set.

3.2 A new dynamic algorithm

To modify the above technique to solve MEB, we propose two ideas. First, instead of the static constant-factor algorithm for minimum bounding box, we use Bădoiu and Clarkson's algorithm for MEB (see Section 2.1) as the starting point. A variant of Bădoiu and Clarkson's algorithm fits nicely here: instead of picking p_j to be the point in P farthest from $c(\mathrm{MEB}(\{p_1, \ldots, p_{j-1}\}))$, we look at the $\alpha|P|$ farthest points from $c(\mathrm{MEB}(\{p_1, \ldots, p_{j-1}\}))$ and pick p_j to be one of them at random with uniform probability. Secondly, instead of returning the union of the coresets found, we use Agarwal and Sharathkumar's blurred ball cover concept to get a good approximation factor. In order to satisfy property (P2), the key is to add all points from previous coresets found into the current point set. The precise details are given in pseudocode form in Algorithms 1–3.

The set P at the root level is initialized with $\widehat{K}_P = \emptyset$. Let u be the maximum number of levels of recursion. Note that $|\widehat{K}_P| \leq O(u/\varepsilon)$ at all levels. For $|P| \gg u/\varepsilon$, note that $|R|$ is a fraction less than $|P|$ if we make the constants α and δ sufficiently small (relative to ε). Thus, for c sufficiently large, u is bounded logarithmically in n. Let $\mathcal{K} = \langle K_1, \ldots, K_u \rangle$ denote the sequence of coresets K_P over all currently active point sets P, arranged from the root level to the last level. Let $B_i = \mathrm{MEB}(K_i)$. Then property (P3) is satisfied because of Bădoiu

Algorithm 1 P.preprocess()

if $|P| < c \log n$ **then**
  $K_P \leftarrow P$ and return
end if
$Q \leftarrow P$
$p_0 \leftarrow$ random point of P
for $j = 1, \ldots, \lceil 2/\varepsilon \rceil$ **do**
  $A_j \leftarrow$ the $\alpha|P|$ farthest points of Q from $c(\text{MEB}(\{p_0, \ldots, p_{j-1}\}))$
  $Q \leftarrow Q - A_j$
  $p_j \leftarrow$ a random point of A_j
end for
$K_P \leftarrow \{p_0, \ldots, p_{\lceil 2/\varepsilon \rceil}\}$ {an ε-coreset of Q by Bădoiu and Clarkson}
$\widehat{K}_R \leftarrow \widehat{K}_P \cup K_P$ {$\widehat{K}_P$ is a union of coresets at earlier levels}
$R \leftarrow (P - Q) \cup \widehat{K}_P$ {remember to add earlier coresets $\widehat{K}_P$ to the next level}
R.preprocess()
P.counter $\leftarrow \delta|P|$

Algorithm 2 P.delete(p), where $p \in P - \widehat{K}_P$

if $|P| < c \log n$ **then**
  remove p from P, reset $K_P \leftarrow P$, and return
end if
remove p from P
P.counter $\leftarrow P$.counter $- 1$
if P.counter $= 0$ or $p \in K_P$ **then**
  P.preprocess() {rebuild all sets after current level}
end if
if $p \in R$ **then**
  R.delete(p)
end if

Algorithm 3 P.insert(p)

if $|P| < c \log n$ **then**
  insert p into P, reset $K_P \leftarrow P$, and return
end if
insert p into P
P.counter $\leftarrow P$.counter $- 1$
if P.counter $= 0$ **then**
  P.preprocess() {rebuild all sets after current level}
end if
R.insert(p)

and Clarkson's algorithm. The trick of inserting coresets at earlier levels to the current set ensures property (P2). We can then use Theorem 1 to infer that $B = \text{MEB}(\bigcup_{i=1}^{u} B_i)$ is a 1.22-approximation to $\text{MEB}(P)$ for a sufficiently small ε.

Instead of applying an exact algorithm to compute B, it is better to first compute a $(1+\varepsilon)$-approximation B_i' to $\text{MEB}(K_i)$ for each i, and then return a $(1+\varepsilon)$-approximation to $\text{MEB}(\bigcup_{i=1}^{u} B_i')$. (Note that every K_i has size $O(1/\varepsilon)$, except for the last set, which has size $O(\log n)$.) The latter can be done by a known approximation algorithm of Kumar, Mitchell, and Yildirim [10], which generalizes Bădoiu and Clarkson's algorithm for sets of balls. The time required is $O(du) = O(d \log n)$ (we ignore dependencies on ε from now on). It can be checked that the proof of Theorem 1 still goes through with B_i replaced by B_i', since the hemisphere property is still satisfied "approximately" for B_i'.

The **for** loop in P.preprocess() takes $O(dn)$ time for constant ε. Thus, the total preprocessing time is bounded by a geometric series summing to $O(dn)$. Space is $O(dn)$ as well. In the pseudocode for P.delete(), although the cost of the call to P.preprocess() is $O(d|P|)$, it can be shown [9] that the probability of deleting an anchor point $p \in K_i$ is $O(1/|P|)$ at any fixed level. Excluding the cost of computing B, the analysis of the expected amortized update time is essentially the same as in Chan's paper [9] and yields $O(d \log n)$. (The randomized analysis assumes that the update sequence is oblivious to the random choices made by the algorithm.) We conclude:

Theorem 2. *A factor-1.22 approximation of the* MEB *of points in* $\mathbb{R}^d$ *can be maintained with an algorithm that takes preprocessing time* $O(dn \log n)$*, uses space* $O(dn)$ *and takes expected amortized time* $O(d \log n)$ *for the updates.*

4 Final Remarks

Agarwal and Sharathkumar [2] have given an example showing that their streaming algorithm has approximation factor strictly greater than their $(1+\sqrt{2})/2$ lower bound. Thus, if their lower bound is the right answer, a different streaming algorithm would be required. It would be interesting to investigate other high-dimensional geometric problems besides MEB in the streaming model. For example, before the recent developments on MEB, Chan [8] had given a $(5+\varepsilon)$-factor streaming algorithm for the smallest enclosing cylinder problem with $O(d)$ space. Can the new techniques help in improving the approximation factor further?

On the dynamic MEB problem, the $(1+\sqrt{2})/2$ lower bound on the approximation factor is not applicable, and the possibility of a $(1+\varepsilon)$-factor algorithm with $d^{O(1)} \log n$ update time and $d^{O(1)} n$ space has not been ruled out. Also, $O(d \log n)$ may not necessarily be the final bound on the update time. For example, Chan [9] described an $O(1)$-factor dynamic algorithm with $O(d)$ expected amortized update time for the smallest enclosing cylinder problem in the word RAM model.

References

1. P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51:606–635, 2004.
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 Sympos. Comput. Geom.*, pages 1–10, 2007.
4. G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38:91–109, 2001.
5. J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformations. *J. Algorithms*, 1:301–358, 1980.
6. M. Bădoiu and K. L. Clarkson. Smaller core-sets for balls. In *Proc. 14th ACM-SIAM Sympos. Discrete Algorithms*, pages 801–802, 2003.
7. M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proc. 34th ACM Sympos. Theory Comput.*, pages 250–257, 2002.
8. T. M. Chan. Faster core-set constructions and data stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35:20–35, 2006.
9. T. M. Chan. Dynamic coresets. *Discrete Comput. Geom.*, 42:469–488, 2009.
10. P. Kumar, J. S. B. Mitchell, and E. A. Yildirim. Approximating minimum enclosing balls in high dimensions using core-sets. *ACM J. Experimental Algorithmics*, 8:1.1, 2003.
11. H. Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. In *Proc. 16th European Sympos. Algorithms*, volume 5193 of *Lect. Notes in Comput. Sci.*, pages 817–829. Springer-Verlag, 2008.
12. H. Zarrabi-Zadeh and T. M. Chan. A simple streaming algorithm for minimum enclosing balls. In *Proc. 18th Canad. Conf. Comput. Geom.*, pages 139–142, 2006.