*Journal of Information & Optimization Sciences*
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
DOI : 10.47974/JIOS-2146

# Approximation algorithms for the minimum enclosing ball problem in high dimensions

Mahdi Imanparast
*Department of Computer Science*
*University of Bojnord*
*Bojnord 9453155111*
*North Khorasan*
*Iran*

**Abstract**

We investigate the minimum enclosing ball problem for given $n$ points in fixed dimension $d$. We present three new approximation algorithms to the problem: a $\sqrt{d}$-approximation algorithm, a 2-approximation algorithm, and a 1.619-approximation algorithm, all running in $\mathcal{O}(nd)$ time. A critical feature of our algorithms is their simplicity, allowing for rapid computation without recourse to exhaustive optimization techniques. While they may not always yield the minimum enclosing ball, they provide a viable solution that is often sufficient for practical applications, especially when working with large datasets.

***Subject Classification:*** *68W25, 65Dxx.*

***Keywords:*** *Minimum enclosing ball, Computational geometry, Approximation algorithms, High dimensions, Time complexity.*

## 1. Introduction

The minimum enclosing ball (MEB) problem, also known as the minimum enclosing sphere, is a fundamental problem in computational geometry. This problem is crucial not only for theoretical analysis but also for practical applications across various fields, including computer graphics, machine learning, robotics, and spatial data analysis. Efficient computation of the MEB can significantly impact performance metrics such as resource allocation in facility location, scene representation in visual computing, and collision detection in virtual environments. The MEB problem originated in 1857, when the mathematician Sylvester [1] initially posed the problem of finding the smallest

*ORCID-ID:* 0000-0003-1674-2247

*E-mail:* m.imanparast@ub.ac.ir

© Taru Publications

disk that would contain a set of $n$ points in two dimensions, which we now refer to as the minimum enclosing disk (MED). Although many attempts have been made to solve this problem, the best outcome came in 1984 when Megiddo [2] proposed a linear-time algorithm for fixed-dimensional linear programming that was a direct approach to the MEB problem and showed that efficient solutions were possible, given certain conditions. Several attempts have been made to compute a near-optimal minimum enclosing sphere in three-dimensional space [3] and to compute the minimum enclosing circle for a convex polygon in $\mathcal{O}(n \log n)$ time [4]. However, the best results were obtained with a randomized algorithm proposed by Welzl [5] for computing the MED of a finite set of points in a plane, which had expected time $\mathcal{O}(n)$ and was based on a linear programming algorithm. Recently, Smolik and Skala [6] proposed modifications of Welzl's algorithm to speed up the process, and reduced the run time of the original Welzl's algorithm by preprocessing the input set of points. The extension of the minimum enclosing disk problem into three-dimensional space introduces additional complexities and challenges, particularly concerning degeneracies and numerical stability. Recent efforts have sought to tackle some of these degeneracies using advanced linear programming methods; however, these often incur increased computational costs.

In high dimensions, where the curse of dimensionality poses significant obstacles, approximate algorithms have garnered attention as viable alternatives. On the other hand, studying this type of problem in higher $d$-dimensional spaces for a fixed dimension $d$ has many applications both theoretically and practically [7]. The initial efforts to address the minimum enclosing ball problem for a collection of points in a higher dimension space were undertaken in the previous work with Badoiu, Har-Peled, and Indyk [8]. They demonstrated that a $(1+\varepsilon)$-approximation to the MEB can be calculated in $\mathcal{O}(nd/\varepsilon + (1/\varepsilon)^5)$ time. Afterwards, Kumar, Mitchell, and Yildirim [9] produced a $(1+\varepsilon)$-approximation based on a core-set of size $\mathcal{O}(1/\varepsilon)$. In line with the previous contributions, Fischer, Gartner, and Kutz [10] and Fischer, and Gartner [11] also investigated the minimum enclosing ball problem for a collection of points in a higher dimension space. They pointed out that the linear randomized Welzl algorithm can fail to compute the minimum ball that can cover a collection of points. Nielsen and Nock [12] offered a $(1+\varepsilon)$-approximation with a running time of $\mathcal{O}(n \log(1/\varepsilon))$ for a collection of $n$ points in the plane. Zarrabi-zadeh and Chan [13] presented a simple $3/2$-approximation algorithm with a running time of $\mathcal{O}(dn)$ when computing the MEB on a set of points in higher dimensional space for the stream data.

In this study, we propose a set of three simple approximation algorithms, namely a $\sqrt{d}$-approximation, a $2$-approximation, and a $1.619$-approximation, to calculate the MEB of the collection of $n$ points on the $d$-dimensional space with a running time of $\mathcal{O}(nd)$. The organization of this paper is as follows. In Section 2, we describe the $\sqrt{d}$-approximation algorithm including an analysis of its memory consumption and running time, as well as its approximation factor. In Section 3, we describe a $2$-approximation algorithm and analyze its running time

and approximation factor. In Section 4, we present a $1.619$-approximation and its related analysis, and Section 5 is the conclusion of the paper.

## 2. The First Algorithm

In this section, we introduce a new approximation algorithm for finding the minimum enclosing ball (MEB) of a set of $n$ points in a $d$-dimensional space. The minimum enclosing ball is defined as the smallest sphere, in terms of radius, which contains the points of the set. This exact solution can be computationally intensive, especially in high-dimensional spaces, so in this section we provide a $\sqrt{d}$-approximation algorithm for this problem.

**Algorithm 1**

**Input:** A point set $P$ in $\mathbb{R}^d$.
**Output:** An approximate solution $\tilde{S}$.
1: Find the axis-aligned minimum bounding box $B(P)$ of point set $P$;
2: $\ell \leftarrow$ Find the length of the longest side of the bounding box $B(P)$;
3: $O \leftarrow$ Compute center of $B(P)$;
4: Draw a sphere $S$ with radius $r = \ell/2$ centered at $O$;
5: Draw a sphere $S_1$ with radius $\sqrt{d}r$ centered at $O$;
6: **if** (all points are covered by sphere $S$)
7: $\tilde{S} \leftarrow S$;
8: **else**
9: $\tilde{S} \leftarrow S_1$;
10: Output $\tilde{S}$;

The algorithm begins by computing the axis-parallel minimum bounding box $B(P)$ for the point set $P$. The minimum bounding box, essentially a hyper-rectangle, is defined by the maximum and minimum coordinates of the points along each dimension. Next, the algorithm determines the size of the largest side $\ell$ of the bounding box $B(P)$. This value serves as an important measurement because it captures the largest extent of the data distribution in any dimension. The center $O$ of the bounding box is then computed. This center point is crucial as it will serve as the center of the spheres drawn in subsequent steps. The algorithm creates a sphere $S$ with a radius $r = \ell/2$ centered at $O$. This radius is half the length of the largest side of the bounding box, providing an initial candidate sphere that covers a significant portion of the points distribution. A second sphere $S_1$ is drawn with a larger radius of $\sqrt{d}r$, also centered at $O$. See Figure 1. The factor $\sqrt{d}$ accounts for the dimensionality of the space, effectively scaling the covering sphere to ensure it has a higher probability of encompassing all points, particularly in higher dimensions where the data might be more dispersed. The algorithm checks whether all points are covered by the first sphere $S$. If they are, it returns this sphere as the approximate solution. If any points lie outside $S$, it defaults to returning the larger sphere $S_1$. It should be noted that the

definition of a sphere with radius $r = \ell/2$ does not necessarily cover all the points of the given point set (see Figure 1). Therefore, we should look for a larger sphere $S_1$ that is closer to the optimal solution to the problem. So, in step 5 of the algorithm, a sphere with radius $\sqrt{d}r$ and centered at $O$ is computed, and among these two spheres, one of them that can cover all the given points is returned as the final solution of the algorithm.

2.1 *Time Complexity Analysis of the Algorithm 1*

The time complexity assessment of Algorithm 1 by which we can determine the minimum enclosing ball in d-dimensional space can be divided into several steps. To compute the $B(P)$, we need to determine the extremes (minimum and maximum) of the coordinates along each dimension. This requires iterating through all $n$ points and updating the minimum and maximum values for each of the $d$ dimensions with $\mathcal{O}(nd)$ time. After calculating the length of the sides of the bounding box, we need to find the largest side among them. Since there are $d$ dimensions, this requires a simple linear scan through the array of side lengths in $\mathcal{O}(d)$ time. To find the center $O$ of the bounding box, we need to average the coordinates of the vertices of the bounding box. Again, this requires considering $n$ points to extract their coordinates and computing the average for each dimension in $\mathcal{O}(nd)$ time. To verify that every point is covered by the sphere $S$ with radius $r = \ell/2$ centered at $O$, requires to get the distance of $O$ from each point and check that the distance is less than or equal to $r$. This requires $n$ distance calculations, which are $\mathcal{O}(d)$ for each point and is $\mathcal{O}(nd)$ in general. Similar to step 4, checking if all points are covered by a sphere $S_1$ also requires $\mathcal{O}(nd)$ time for the same reasons. Thus, the overall time complexity is $\mathcal{O}(nd)$.

2.2 *Memory Consumption of the Algorithm 1*

The memory usage of the algorithm for finding the minimum enclosing ball in $d$-dimensional space can be analyzed based on the various components and data structures utilized throughout the algorithm. The algorithm needs to store the $n$ input points, each represented as a vector in $d$-dimensional space. This requires $\mathcal{O}(nd)$ memory. To determine the minimum bounding box $B(P)$, we need to store the minimum and maximum for each dimension in two arrays, each of size $d$ with $\mathcal{O}(d)$ required space. An additional array is needed to store the lengths of the sides of $B(P)$, which also has a length of $d$. The center $O$ of the bounding box is a single point in $d$-dimensional space, requiring $d$ coordinates of storage, or $\mathcal{O}(d)$. The algorithm may use a few additional variables (e.g., for radius calculations) but these require constant space $\mathcal{O}(1)$. Thus, the total memory of the algorithm can be expressed as $\mathcal{O}(nd)$. In fact, the dominant term in memory usage comes from storing the $n$ points in $d$-dimensional space, leading to an overall memory requirement of $\mathcal{O}(nd)$.

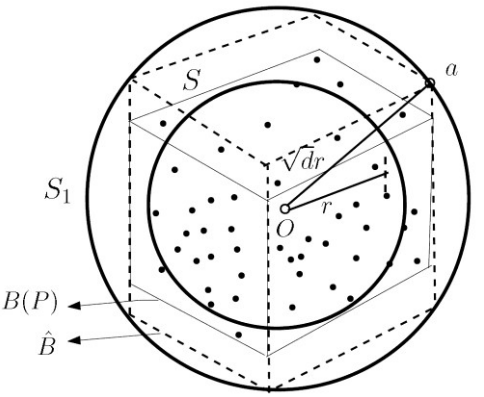



**Figure 1**
**Two spheres $S$ and $S_1$ based on the minimum bounding box $B(P)$ and the cubic box $\hat{B}$ with side length $2r$, respectively.**

2.3 *Approximation Factor Analysis of the Algorithm 1*

We examine the approximation factor of the proposed algorithm in this subsection.

**Theorem 1:** *Let $r^*$ be the radius of the optimal minimum enclosing ball and $r_{\tilde{S}}$ be the radius of the ball that is returned by the proposed algorithm for a set of $n$ points in the $d$-dimensional space. Then, the proposed algorithm has the approximation factor $\sqrt{d}$ and $||r_{\tilde{S}}|| \leq \sqrt{d}||r^*||$.*

***Proof:*** Let $O = (x_1, x_2, \cdots, x_d)$ be the center of the minimum bounding box and $a = (x_1 + r, x_2 + r, \cdots, x_d + r)$ be the farthest point to the center $O$ on the boundary of the equal side box (cubic box) $\hat{B}$. Note that the $\hat{B}$ is a cubic box with a side length $2r$ which encloses all the points (see Figure 1). Then, for $r_1$ the radius of the sphere $S_1$, we have the following relationship between the length of $r_1$ and $r$:

$$||r_1|| = ||a - O|| = \sqrt{(x_1 + r - x_1)^2 + (x_2 + r - x_2)^2 + \cdots + (x_d + r - x_d)^2}$$
$$= \sqrt{r^2 + r^2 + \cdots + r^2} = \sqrt{dr^2} = \sqrt{d}||r||. \quad (1)$$

Therefore,

$$||r_1|| = \sqrt{d}||r||. \quad (2)$$

Let $r_{\tilde{S}}$ be the radius of the final solution sphere $\tilde{S}$. It is obvious that:

$$||r|| \leq ||r_{\tilde{S}}|| \leq ||r_1||. \quad (3)$$

So, according to relation (2):

$$||r|| \leq ||r_{\tilde{S}}|| \leq \sqrt{d}||r||. \quad (4)$$

Then, according to the definition of the initial sphere $S$, we have:

$$||r|| \leq ||r^*||. \quad (5)$$

Now, according to the relations (4) and (5):

$$||r|| \leq ||r_{\tilde{S}}|| \leq \sqrt{d}||r^*||. \quad (6)$$

Therefore,

$$||r_{\tilde{S}}|| \leq \sqrt{d}||r^*||. \quad (7)$$

This means that $r_{\tilde{S}}$ is at most $\sqrt{d}$ times the radius $r^*$ of the optimal sphere $S^*$, and this complete the proof.

## 3. The Second Algorithm

We present a new approximation algorithm with an approximation factor of 2 for the smallest enclosing ball problem.

**Algorithm 2**

**Input:** A point set $P$ in $\mathbb{R}^d$.
**Output:** An approximate solution for the minimum enclosing ball.
1: Compute the axis-parallel minimum bounding box $B(P)$ for a point set $P$;
2: $O \leftarrow$ Compute the center of the bounding box $B(P)$;
3: Calculate radius $r = \sqrt{\left(\frac{L_1}{2}\right)^2 + \left(\frac{L_2}{2}\right)^2 + \cdots + \left(\frac{L_d}{2}\right)^2}$, where $L_i$ is the side length of the MBB in dimension $i$;
4: Return the sphere $S$, centered at $O$ and radius $r$.

Regarding Algorithm 2, we have:

**Theorem 2:** *Algorithm 2 for the problem of finding the minimum enclosing ball has approximation factor* 2 *and has a running time of* $O(dn)$.

***Proof:*** Computing the minimum bounding box takes $O(dn)$ time. The time to compute the center of the minimum bounding box will take $O(d)$ time, and considering the constant time required for steps 3 and 4, the total running time of the algorithm will be $O(dn)$. Let $r^*$ denote the radius of the optimal MEB for the set of points $P$ with center $O_m$. We need to show that the radius $r$ of the output sphere $S$ produced by the algorithm satisfies $r \leq 2r^*$ (See Figure 2). First we show that $d = ||p_i p_k|| \leq r$. We have $D^2 \approx d^2 + \ell^2$, so, $d^2 \approx D^2 - \ell^2$. Since $D \leq r^*$ and $\ell = r^*\theta$, we have $d^2 \leq (r^*)^2 - (r^*)^2\theta^2$. This means that $d^2 \leq (r^*)^2(1 - \theta^2)$. Since $0 \leq \theta \leq \frac{\pi}{4}$, therefore $d^2 \leq (r^*)^2$, and so

$$d \leq r^*. \quad (8)$$

On the other hand, according to the relation (8), $||p_l p_j|| \leq r^*$ and $||p_i p_k|| \leq r^*$. Moreover, we have $||p_j O|| + ||O p_i|| \leq 2r^*$ (see Figure 2). Therefore,

$$||p_l p_k|| = ||p_l p_j|| + ||p_j O|| + ||O p_i|| + ||p_i p_k|| \leq 4r^*. \quad (9)$$

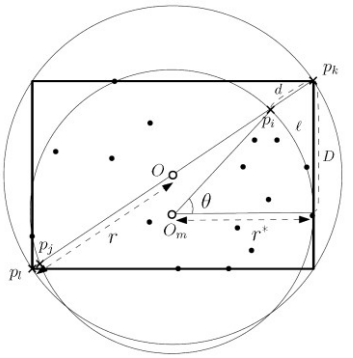



**Figure 2**

**An optimal ball with the center $O_m$ and radius $r^*$ and a ball with the center $O$ and radius $r$ were returned by the Algorithm 2.**

However, because $p_l$ and $p_k$ are two corner points of the minimum bounding box, we have:

$$||p_l p_k|| = ||p_l O|| + ||O p_k||. \tag{10}$$

Hence, according to relations (9) and (10):

$$||p_l p_k|| = ||p_l O|| + ||O p_k|| \leq 4r^*. \tag{11}$$

Since $r = ||p_l O|| = ||O p_k||$. Therefore $2||O p_k|| \leq 4r^*$ and so $||O p_k|| \leq 2r^*$. Thus

$$r \leq 2r^*. \tag{12}$$

and this completes the proof.

## 4. The Third Algorithm

We give a new and improved approximation algorithm with an approximation factor of $1.619$ for the MEB problem.

---

**Algorithm 3**

---

**Input:** A point set $P$ in $\mathbb{R}^d$.

**Output:** An approximate solution $\tilde{S}$.

1: Compute the axis-parallel minimum bounding box $B(P)$ for the input point set $P$;
2: $\ell \leftarrow$ Find the length of the largest side in $B(P)$;
3: $O \leftarrow$ Compute center of the bounding box $B(P)$;
4: Draw a sphere $S$ with radius $r = \ell/2$ centered at $O$;
5: If sphere $S$ covers all the points, send it as output $\tilde{S} \leftarrow S$. Otherwise, go to step 6;
6: $S' \leftarrow S$, $O' \leftarrow O$, $r' \leftarrow r$;
7: While ($P \neq \phi$);

8: **begin**
9: Find the points inside sphere $S'$ and place them in $U$ and set $P = P - U$;
10: Find a point $b \in P$ that is the farthest point from $O'$;
11: Compute distance $d$ between $O'$ and $b$ by relation $d = (||b - O'|| - r')/2$;
12: Draw a sphere $S'$ with radius $r' = r' + d$ and center $O = O' + (b - O')d/||b - O'||$;
13: If sphere $S'$ covers all the points, send it as output $\tilde{S} \leftarrow S'$. Otherwise, go to step 7;
14: **end of while**
15: Output $\tilde{S}$;

By adjusting the center and radius iteratively, the method reduces the distance needed to cover remaining points and hence can more closely approximate the optimal solution. The iterative approach allows for the re-evaluation of the sphere size and its location based on the actual distribution of points. An example of running Algorithm 3 on a set of points is shown in Figure 3. Regarding the Algorithm 3, we have.

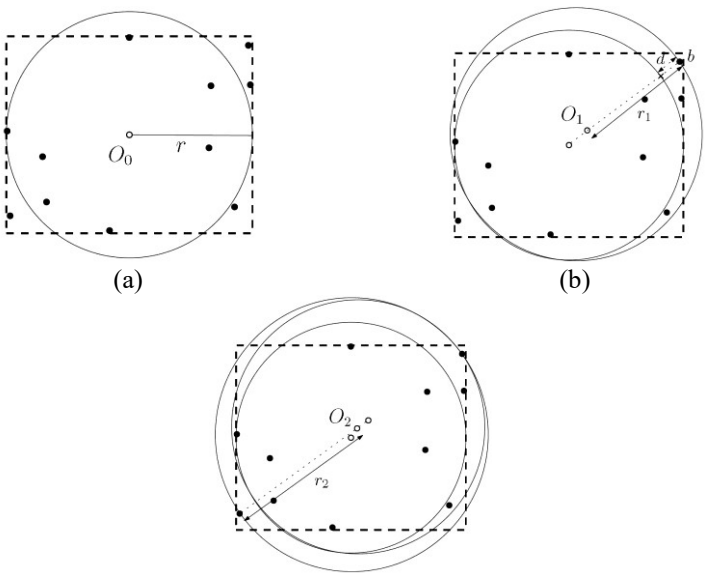



**Figure 3**

**An instance of running the Algorithm 3. (a) Initial sphere $S$ with radius $r$ and center $O = O_0$. (b) Sphere created in the first iteration of the loop with radius $r_1$ and center $O_1$. (c) Final sphere created in the second iteration of the loop with radius $r_2$ and center $O_2$.**

**Theorem 3:** *The Algorithm 3 for the minimum enclosing ball problem has an approximation factor* $1.619$ *and a running time* $\mathcal{O}(dn)$.

***Proof:*** For the running time of the algorithm, we note that it takes $\mathcal{O}(dn)$ to compute the minimum bounding box, $\mathcal{O}(d)$ to compute the center of the minimum bounding box, and $\mathcal{O}(d)$ for each point and $\mathcal{O}(dn)$ for all points to determine if a point is inside the sphere. The calculations related to the steps inside the while loop require a constant time in total for each step, and considering the loop iterations, which will be less than $n$, the total time of the algorithm will be $\mathcal{O}(dn)$. Let $r^*$ denote the radius of the optimal minimum enclosing ball for the set of points $P$. Let $b$ denote the point that is the farthest point from the center $O$ of the first sphere. It is clear that the greatest distance between $O$ and $b$ occurs when $b$ is a corner point or close to it in an equal sides minimum bounding box (see Figure 4). In this case, consider two vectors $OA$ and $OB$, where the angle between them is $\theta$. Given the length of the arc $\ell = AB = r\theta$, we have $D^2 \approx d^2 + \ell^2 = d^2 + r^2\theta^2$. Since $D \leq r$, $d^2 + r^2\theta^2 \leq r^2$. Therefore,

$$d^2 \leq r^2 - r^2\theta^2 = r^2(1 - \theta^2). \tag{13}$$

Since the maximum value of $d$ occurs when $\theta = \pi/4$, we have:$r^2(0.383)$. Therefore,

$$d \leq 0.619r. \tag{14}$$

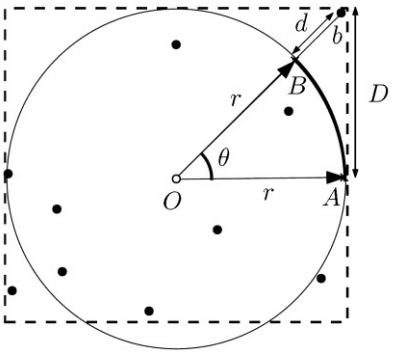



**Figure 4**
**The first sphere is created by Algorithm 3 with radius $r$ and center $O$ along with the farthest point $b$ from the center of the sphere.**

Continuing the loop iterations, we will eventually reach a sphere with a diameter of $2r + 2d$ or radius $r + d$. Therefore, the radius of the sphere resulting from Algorithm 3 will be $r + 0619r = 1.619r$.

Since for the first sphere $S$, we have:

$$r \leq r^* \tag{15}$$

On the other hand, we have:

$$r \leq r_{\tilde{S}} \leq 1.619r \tag{16}$$

Therefore, from two relations (15) and (16), we can conclude:

$$r_{\tilde{S}} \leq 1.619 r^* \tag{17}$$

This shows that Algorithm 3 has an approximation factor of $1.619$ and this completes the proof.

## 5. Conclusions

In this paper, we have presented three novel approximation for computing the minimum enclosing ball problem in $\mathbb{R}^d$. The algorithms are a $\sqrt{d}$-approximation, a $2$-approximation and a $1.619$-approximation with the running time $\mathcal{O}(dn)$. By utilizing the geometric characteristics of axis-aligned minimum bounding boxes, our approaches significantly simplify the complexity typically associated with minimum enclosing ball problem computations. Moreover, the simple design of our algorithms facilitate easy implementation and adaptability to various contexts and data distributions. Future work can explore more refinements to enhance accuracy, as well as potential extensions to more complex geometries or dynamic datasets and varying dimensionality.

## References

[1] J. J. Sylvester, "A question in the geometry of situation," *Quarterly Journal of Pure and Applied Mathematics,* vol. 1, no. 1, pp. 79-80 (1857).

[2] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *Journal of the ACM* (JACM), vol. 31, no. 1, pp. 114-127 (1984).

[3] J. Ritter, "An efficient bounding sphere," In Game Programming Gems, A. Glassner (Ed.), Academic Press, Boston, pp. 301-303 (1990).

[4] S. Skyum, "A simple algorithm for computing the smallest enclosing circle," *Information Processing Letters*, vol. 37, no. 3, pp. 121-125 (1991).

[5] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," In New Results and New Trends in Computer Science (LNCS 555), H. Maurer (Ed.), Springer- Verlag, pp. 359-370 (1991).

[6] M. Smolik, and V. Skala, "Efficient Speed-Up of the Smallest Enclosing Circle Algorithm," *Informatica*, vol. 33, no. 3, pp. 623-633 (2022).

[7] M. Imanparast, S. N. Hashemi, and A. Mohades, "Efficient approximation algorithms for point-set diameter in higher dimensions," *Journal of Algorithms and Computation*, vol. 51, no. 2, pp. 47-61 (2019).

[8] M. Bădoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," In Proc. 34th annual ACM symposium on Theory of computing, pp. 250-257 (2002).

[9] P. Kumar, J. S. Mitchell, and E. A. Yildirim, "Approximate minimum enclosing balls in high dimensions using core-sets," *Journal of Experimental Algorithmics* (JEA), vol. 8, pp.1-1 (2003).

[10] K. Fischer, B. Gärtner, and M. Kutz, "Fast smallest enclosing- ball computation in high dimensions," In Proc. 11th Annual European Symposium on Algorithms, LNCS 2832. Springer-Verlag, pp. 630-641 (2003).

[11] K. Fischer, and B. Gärtner, "The smallest enclosing ball of balls: combinatorial structure and algorithms," In Proc. 19th annual symposium on Computational Geometry, 292-301 (2003).

[12] F. Nielsen, and R. Nock, "A fast deterministic smallest enclosing disk approximation algorithm," *Information Processing Letters*, vol. 93, no. 6, pp. 263-268 (2005).

[13] H. Zarrabi-Zadeh, and T. M. Chan, "A Simple Streaming Algorithm for Minimum Enclosing Balls," In Proc. 18th Canadian Conference on Computational Geometry, pp. 139-142 (2006).

*Received March, 2025*