Improved bounds for Smallest Enclosing Disk Range Queries

Sankalp Khare* Jatin Agarwal† Nadeem Moidu‡ Kannan Srinathan§

Abstract

Let S be a set of n points in the plane. We present a method where, using $O(n \log^2 n)$ time and space, S can be pre-processed into a data structure such that given an axis-parallel query rectangle q, we can report the radius of the smallest enclosing disk of the points lying in $S \cap q$ in $O(\log^6 n)$ time per query.

1 Introduction

The *range-aggregate query* problem is a variant of range searching wherein the goal is to preprocess a set of points, S, into a data structure such that given a query region q, the value of an *aggregate function* f, over the region $S \cap q$, can be computed efficiently. When compared to standard range queries, a range-aggregate query provides a more informative summary of the output. Range aggregate query problems have been the focus of much work in recent literature [15] [8] [13] [14] [4] [3].

Aggregate functions can be numeric, for example *count*, *sum* (of weights), etc. or geometric, such as *width*, *closest/farthest pair*, *maximal/dominating chain* etc. Geometric aggregate functions are, in general, non-decomposable, i.e., the desired result $f(S \cap q)$ cannot be derived efficiently from the results obtained by applying f on partitions of $S \cap q$. This calls for the development of more sophisticated techniques to answer geometric range aggregate queries.

All previously known non-trivial results for the *smallest enclosing disk* range-aggregate query problem were approximation versions [11] [10] . Brass et al. [3] presented the first *exact*, non-trivial solution. Their solution requires $O(n \log^2 n)$ preprocessing time and $O(n \log^2 n)$ space to build a data structure which answers queries in $O(\log^9 n)$ time. We improve upon their result, presenting a method that utilizes the same amount of time and space for preprocessing, but answers queries in $O(\log^6 n)$ time.

*Center for Security, Theory & Algorithmic Research (CSTAR), International Institute of Information Technology, Hyderabad (IIIT-H), India, sankalp.khare@research.iiit.ac.in

†CSTAR, IIIT-H, India, jatin.agarwal@research.iiit.ac.in

‡CSTAR, IIIT-H, India, nadeemoidu@gmail.com

§CSTAR, IIIT-H, India, srinathan@iiit.ac.in

2 Preliminaries

We assume that the points in S are in general position. Let q be an orthogonal query range of the form $[a_x, b_x] \times [a_y, b_y]$, where $a_x < b_x$ and $a_y < b_y$.

Proposition 1 *Given a set of points, S, in the plane, the smallest enclosing disk,* $\mathrm{sed}(S)$*, is the minimal radius disk such that every point $p \in S$ lies on or within the boundary of* $\mathrm{sed}(S)$*.*

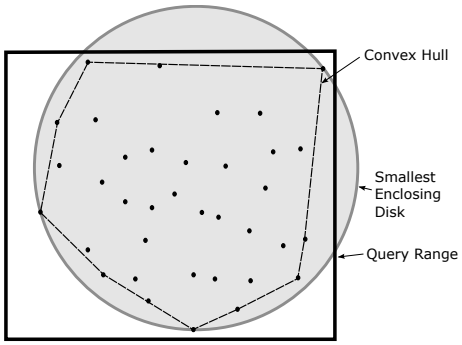

Figure 1: Smallest Enclosing Disk and Convex Hull in a query range

3 Previous Work

In Section 5.1 of [3], the authors demonstrate how, for a set of points in the plane, the problem of computing the radius of the smallest enclosing disk can be transformed to the problem of finding the minimum vertical distance between a convex polyhedron and a paraboloid in the dual space. The methods used to achieve this transformation are well studied in the literature (Section 5.7 of [12]), but for completeness we will outline the procedure here.

The remainder of section 3 covers relevant parts of the solution by Brass et al. [3].

3.1 The lifting map:

A lifting map *lifts* points in two dimensions into the three dimensional space by assigning them a z coordinate which is a function of their x and y coordinates. We define our lifting map so that a point

in $\mathbb{R}^2$ is mapped to a point on the paraboloid $P : z = x^2 + y^2$ in $\mathbb{R}^3$:

$$z = f(x, y) = x^2 + y^2$$

$$p = (x, y) \;\mapsto\; p_\uparrow = (x, y, x^2 + y^2)$$

3.2 The duality transform

A duality transform defines a mapping between any two sets of parameterized geometric objects. It provides an alternative way to view the same information. We employ the following duality transform, which maps a non-vertical plane H to a point H^*, in $\mathbb{R}^3$:

$$H : z = ax + by + c \;\mapsto\; H^* : (a/2, b/2, c)$$

Let C be a circle in $\mathbb{R}^2$, centered at (a, b) and with radius r. Let H_C be the non-vertical plane defined as:

$$H_C : z = 2ax + 2by + r^2 - a^2 - b^2$$

A point $p : (x, y)$ in $\mathbb{R}^2$ lies on, inside or outside the circle C if and only if the point $p_\uparrow : (x, y, x^2 + y^2)$ lies on, below or above H_C, respectively.

3.3 The Dual Problem

Using the lifting map, our point set S is mapped to the set $S_\uparrow$ defined as:

$$S_\uparrow = \{p_\uparrow \,|\, p \in S\}$$

Let C be a circle with center (a, b) and radius r, such that it encloses all points in the set S. Then all points in $S_\uparrow$ lie on or below the plane H_C. Define the plane H'_C:

$$H'_C : z = 2ax + 2by - a^2 - b^2$$

Clearly H'_C is (a) parallel to H_C, and (b) tangent to the paraboloid P. Also, the vertical distance between the two planes, H_C and H'_C is r^2.

If C were the smallest enclosing disk for the set S, then either two or three points in S would lie on the boundary of C and their lifted counterparts in $S_\uparrow$ would lie on H_C, i.e., either an edge (in the two-point scenario) or a face (in the three-point scenario) of the *upper* convex hull of $S_\uparrow$ would lie on H_C (since all points in $S_\uparrow$ lie below H_C, in the negative z-direction). This brings us to the following observation:

Observation 1 *The square of the radius of* $\mathrm{sed}(S)$ *is the vertical distance between two parallel planes H and H', such that*

1. *all points of $S_\uparrow$ are on or below H,*
2. *H contains either a face or an edge of the upper convex hull (positive z-direction) of $S_\uparrow$, and*
3. *H' is tangent to the paraboloid P.*

$$G^* = \{H^* \mid H \text{ is a non-vertical plane on or above the convex hull of } G\}$$

The following observation then holds:

Observation 2 *For a set of points G in $\mathbb{R}^3$, the set G^* is convex and unbounded in the positive z-direction.*

For a set P' defined as follows,

$$P' = \{H^* \mid H \text{ is a non-vertical plane on or below the paraboloid } P\}$$

Let P^* be the boundary of P'. Then we can make the following observation, similar to Observation 2:

Observation 3 *The set P' is convex and unbounded in the negative z-direction, and P^* is the paraboloid $z = -(x^2 + y^2)$.*

We now have the following geometric constructs:

- $S_\uparrow$, in the primal space, is the set of points obtained by lifting the input point set S.
- $S^*_\uparrow$, in the dual space, is the set of points obtained by applying the duality transform on $S_\uparrow$.
- P^*, in the dual space, is the paraboloid obtained by applying the dual transform on the paraboloid P.

We define a set of points B^*, in the dual space, as follows:

$$B^* = \{H^* \mid H \text{ is a non-vertical plane containing some face of the upper convex hull of } S_\uparrow\}$$

Furthermore, we define $\mathcal{B}^*$ as follows:

$$\mathcal{B}^* = \{p^* \mid p^* \text{ is a point in the dual space which lies in the region } \textit{vertically above} \text{ the lower convex hull of } B^*\}$$

In other words, all points in $\mathcal{B}^*$ project down (negative z-direction) to some point on the lower convex hull of B^*. Clearly, $\mathcal{B}^*$ is a convex polyhedron unbounded in the positive z-direction, such that (a) $\mathcal{B}^*$ is fully contained within $S^*_\uparrow$, and (b) $\mathcal{B}^*$ and P^* are disjoint.

3.4 Finding the radius of the smallest enclosing disk

Consider the planes H and H', as defined in Observation 1. In the dual space, the point H^*, obtained by applying the duality transform on H, lies on the boundary of $S^*_\uparrow$, and the point $(H')^*$ lies on the paraboloid P^*. Since H must contain either an edge or a face of the upper convex hull of $S_\uparrow$, this implies that H^* is either an edge or a vertex of the lower convex hull of B^*, in other words, H^* is either an edge or a vertex of $\mathcal{B}^*$. Thus we have the following:

Observation 4 *Let S be a set of n points in the plane. The radius of* $\text{sed}(S)$ *is equal to the minimum vertical distance between $\mathcal{B}^*$ and P^*.*

For a point set S and query region q in a standard range tree, $S \cap q$ can be expressed as the union of the points rooted at canonical nodes $v_1, v_2, \ldots, v_m$ which fall in the desired range. For all canonical nodes v_i, let $S(v_i)$ be the set of points rooted at the node. We define $S_\uparrow(v_i)$, $S^*_\uparrow(v_i)$ and $\mathcal{B}^*(v_i)$, as described above. For the disk to contain all points in $S \cap q$, the associated plane H must be such that the points in $S_\uparrow(v_i)$, for *all* canonical nodes, lie on or below it. In other words, the plane H associated with $\text{sed}(S \cap q)$ must lie on or above the convex hull of the *union* of all $S_\uparrow(v_i)$. In the dual space equivalent, what this means is that the point H^* must lie on the boundary of the *intersection* of all $S^*_\uparrow(v_i)$. By definition of $\mathcal{B}^*$, this implies that H^* must be a point on the intersection of all $\mathcal{B}^*(v_i)$.

Thus, the problem of computing $\text{sed}(S \cap q)$ is equivalent to that of computing the minimum vertical distance between

- the paraboloid P^*, and
- the intersection of the convex polyhedra $\mathcal{B}^*(v_1), \mathcal{B}^*(v_2), \ldots, \mathcal{B}^*(v_m)$. We call it $\mathcal{B}^*(S \cap q)$.

4 Our Solution

A standard two dimensional range tree partitions q into $O(\log^2 n)$ rectangular sub-regions. The points in each region, via lifting and duality transforms, are mapped to a convex polyhedron in the dual space. To construct the polyhedron $\mathcal{B}^*(S \cap q)$ the authors, in [3], take the intersection of all these $O(\log^2 n)$ polyhedra.

We propose a solution that allows us to discard a large set of canonical nodes in the query range while preserving nodes whose points contribute to the smallest enclosing disk. More precisely, our solution reduces the number of canonical nodes (and corresponding polyhedra $\mathcal{B}^*(v_i)$) under consideration to $O(\log n)$. This reduces the search-space, and therefore query time, significantly.

5 Data Structures and Query Algorithm

We use a modified version of the two dimensional range tree, described in [9], to partition the orthogonal range into a grid, as shown in Figure 2. The idea used is similar to the one used in [2] to enhance kinetic *kd*-trees.

Given a point set P, we first construct a tree T_x, which is a one-dimensional range tree, on the x co-ordinates of the points. For any node v of this tree, the set of all points rooted at v is called the *canonical subset* of v. We denote the canonical subset of a node v by $P(v)$. The standard two-dimensional range

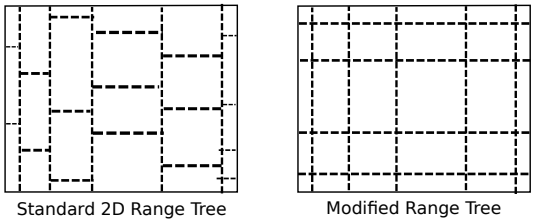

Figure 2: Space partitioning by the standard v/s modified range tree

tree associates each node v with a secondary tree $T_y(v)$, which is a balanced binary search tree built on the y co-ordinates of the points in $P(v)$. Our data structure differs in the method of building these secondary trees.

We build a one-dimensional range tree, T_y, based on the y co-ordinates of the entire point set P. We call this the *template tree*. For each node v, we reduce the template tree into a structure that we call the *contracted tree* for the set $P(v)$. To do so, we perform the following two operations on the template tree T_y (Figure 3):

- First, all subtrees that do not have a leaf in $P(v)$ are removed.
- Subsequently all nodes that have a single child are *contracted*, i.e., the child of such a node is directly connected to its parent, and the node itself is removed. This process is carried out recursively until no nodes with a single child remain.

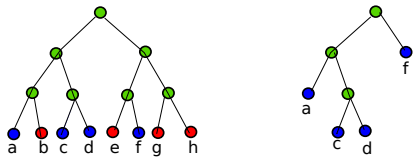

Figure 3: Template tree and Contracted tree for nodes a, c, d and f.

Notice that every range associated with a node of such a contracted tree is also present in T_y, as we are merely eliminating ranges that do not correspond to points in $P(v)$. Since each range in T_y corresponds to a horizontal section of the plane, the contracted tree for any node v represents a subset of these horizontal sections. By using the space partitioning produced by T_y, we can partition the space into a tiled grid, such that all space partitions induced by the contracted secondary trees will automatically be aligned with this grid. This produces the tiled space partitioning shown in Figure 2.

Consider the tiled alignment generated by the modified range tree. For each query we can identify blocks

Algorithm 1 Finding candidate blocks in the top-right region

Initialize: $B \leftarrow$ empty list
Set: $b_x \leftarrow$ block enclosing the point with max x value
Set: $b_y \leftarrow$ block enclosing the point with max y value
1: B.append(b_x)
2: **while** $b_x \neq b_y$ **do**
3: **if** $\exists\ b_{x'} \mid b_{x'}$ is non-empty and lies vertically above b_x, in the same column **then**
4: $b_x \leftarrow b_{x'}$
5: **else**
6: $b_x \leftarrow$ left-neighbour of b_x
7: **end if**
8: B.append(b_x)
9: **end while**
10: **return** B (List of Candidate Blocks)

(tiles) that (a) together capture all points on the boundary of the point-set, and (b) therefore must contain the bounding points of $\text{sed}(S \cap q)$. We call these blocks *candidate blocks*. These blocks alone are sufficient for computing both the convex hull and the smallest enclosing disk.

Lemma 2 *The set $\mathcal{C}$ of candidate blocks can be computed in $O(\log n)$ steps, and $|\mathcal{C}| = O(\log n)$.*

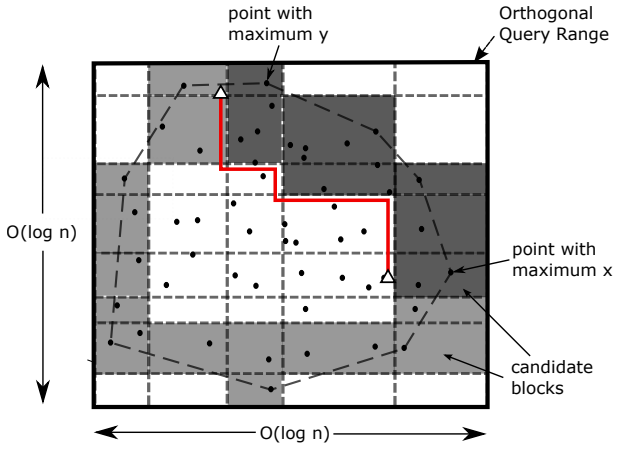

Figure 4: candidate blocks

Proof. The set $\mathcal{C}$ of candidate blocks can be partitioned into four continuous chains based upon the extreme points in each orthogonal direction. We demonstrate how to compute the upper-right quarter (dark shaded blocks in Figure 4):

We begin at the block containing the point with maximum x co-ordinate. We then find the first non-empty block in the positive y direction in the same column. If such a block is found, we move to that block. If not, we move one block to the left and repeat the process until the block enclosing the point with the maximum y co-ordinate is reached. In each step we move either towards the top or towards the left, therefore we may move a maximum of $O(\log n)$ steps in both directions. All non-empty blocks visited in this process constitute the max x to max y chain of candidate blocks. Algorithm 1 describes the process rigorously.

The remaining chains can be computed in a similar manner and the results combined to get the set $\mathcal{C}$. □

In place of the polyhedron $\mathcal{B}^*$ of Section 3.3, we can now use a convex polyhedron $\mathcal{B}^{**}$, defined as the intersection of these $O(\log n)$ candidate blocks/nodes (as opposed to $O(\log^2 n)$ required previously).

At each candidate canonical node v, we store the *hierarchical representation* [6] of the corresponding convex polyhedron $\mathcal{B}_v$, which can be constructed in $O(|\mathcal{B}_v|)$ time and requires an equivalent amount of space. Therefore the entire tree requires $O(n \log^2 n)$ time and space.

Using hierarchical representations of polyhedra and the data structures described in Section 5.2 of [3], we can compute the distance between the paraboloid P^* and a convex polyhedron A in its hierarchical representation, $hier(A)$, in $O(\log |A|)$ time.

Lemma 3 (Eppstein, 1992. [7], as given in [3]) *Given m convex polyhedra represented by their hierarchical representations, we can optimize any given objective function over their common intersection in $O(\gamma \cdot m^3 \log^2 n)$ time, provided that the elementary problem of optimizing the function over one convex polyhedron can be done in $O(\gamma)$ time.*

We have $m = O(\log n)$ polyhedra and $\gamma = O(\log n)$, as explained above, therefore the minimum vertical distance between the paraboloid P^* and the convex polyhedron $\mathcal{B}^{**} = \bigcap_{i \in \mathcal{C}} \mathcal{B}_i$ can be computed in $O(\log^6 n)$. Thus, the radius of $\text{sed}(S \cap q)$ can also be computed within the same time bound.

Theorem 4 *Let S be a set of points in the plane. We can construct an $O(n \log^2 n)$ size data structure in $O(n \log^2 n)$ time, such that for any axis-parallel query rectangle q, the radius of the smallest enclosing disk for the points lying in $S \cap q$ can be computed in $O(\log^6 n)$ time.*

6 Conclusions and Future Work

In this paper we have shown how a non-decomposable range searching problem that uses range trees can be solved in a much more efficient manner using the range tree variant proposed by Moidu et al. in [9]. There exist other problems to which the same optimization can be applied, i.e. problems wherein the aggregate function

can be computed without examining the points lying in the interior[1] of the query region.

We hope that more such problems will be found and improved upon using the ideas herein.

References

[1] *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010*, 2010.

[2] M. A. Abam, M. de Berg, and B. Speckmann. Kinetic kd-trees and longest-side kd-trees. *SIAM J. Comput.*, 39(4):1219–1232, 2009.

[3] P. Brass, C. Knauer, C. S. Shin, M. Smid, and I. Vigan. Range-aggregate queries for geometric extent problems. In *CATS: 19th Computing: Australasian Theory Symposium*, 2013.

[4] A. S. Das, P. Gupta, A. K. Kalavagattu, J. Agarwal, K. Srinathan, and K. Kothapalli. Range aggregate maximal points in the plane. In M. S. Rahman and S.-I. Nakano, editors, *WALCOM*, volume 7157 of *Lecture Notes in Computer Science*, pages 52–63. Springer, 2012.

[5] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[6] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra - a unified approach. In M. Paterson, editor, *ICALP*, volume 443 of *Lecture Notes in Computer Science*, pages 400–413. Springer, 1990.

[7] D. Eppstein. Dynamic three-dimensional linear programming. *INFORMS Journal on Computing*, 4(4):360–368, 1992.

[8] R. Janardan, P. Gupta, Y. Kumar, and M. H. M. Smid. Data structures for range-aggregate extent queries. In *CCCG*, 2008.

[9] N. Moidu, J. Agarwal, and K. Kothapalli. Planar convex hull range query and related problems. In *CCCG*. Carleton University, Ottawa, Canada, 2013.

[10] Y. Nekrich and M. H. M. Smid. Approximating range-aggregate queries using coresets. In *CCCG* [1], pages 253–256.

[11] F. Nielsen and R. Nock. A fast deterministic smallest enclosing disk approximation algorithm. *Inf. Process. Lett.*, 93(6):263–268, 2005.

[12] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

[13] S. Rahul, H. Bellam, P. Gupta, and K. Rajan. Range aggregate structures for colored geometric objects. In *CCCG* [1], pages 249–252.

[14] S. Rahul, A. S. Das, K. S. Rajan, and K. Srinathan. Range-aggregate queries involving geometric aggregation operations. In N. Katoh and A. Kumar, editors, *WALCOM*, volume 6552 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2011.

[15] R. Sharathkumar and P. Gupta. Range-aggregate proximity detection for design rule checking in vlsi layouts. In *CCCG*, 2006.

[1] the meaning of which is made clear in our exposition above