Smallest enclosing circle centered on a query line segment

Prosenjit Bose [*], Stefan Langerman[†], and Sasanka Roy [‡]

Abstract

Given a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in the plane, we show how to preprocess P such that for any query line segment L we can report in $O(\log n)$ time the smallest enclosing circle whose center is constrained to lie on L . The preprocessing time and space complexity are $O(n \log n)$ and $O(n)$ respectively. We then show how to use this data structure in order to compute the smallest enclosing circle of P whose center is restricted to lie in one of several polygons having a total of m edges, in $O((m + n) \log n)$ time, a significant improvement over previous known algorithms.

1 Introduction

The problem of computing the smallest enclosing circle of a set P of n points in the plane was originally posed in 1857 by Sylvester [11]. Many solutions have appeared in the literature (see [9] or [8] for a brief history of the problem) culminating in the optimal linear time algorithm by Megiddo [7].

In recent years, several constrained variants of this problem have been studied, where restrictions are placed on the location of the center of the smallest enclosing disk. Already in his original paper and as a step towards the general solution, Megiddo [7] studied the situation where the center of the smallest enclosing circle is restricted to lie on a given straight line. Hurtado et al. [4] generalized Megiddo's technique to provide an $O(n + m)$ time algorithm for finding smallest enclosing circle whose center is constrained to lie in the intersection of m linear inequalities.

Bose et al. [1] considered a generalized setting of the problem where the center of the smallest enclosing circle of P is constrained to lie inside a simple polygon of size m. Their algorithm runs in $O((n + m) \log(n + m) + k)$ time, where k is the number of intersections of the boundary of the polygon with the farthest point Voronoi diagram of P. In the worst case, k may be $O(n^2)$. This result was later improved to $O((n + m) \log m)$ by Bose and Wang [2]. In a further generalization of this problem, where r (≥ 1) simple polygons with a total of m vertices are given, locating the center of the smallest enclosing circle of P with its center inside one of the given polygons was studied by Bose and Wang [2]. The time complexity of this version of the problem is $O((m + n) \log n + (n\sqrt{r} + m) \log m + m\sqrt{r} + r^{\frac{3}{2}} \log r)$ and which has further improved to $O(n \log n + m \log^2 n)$ by Roy et al. [10].

The query version of the smallest enclosing circle ($QSEC$) where the center is constrained to lie on a query line was originally posed by Roy et al. [10]. The preprocessing time and space complexity of their algorithm is $O(n \log n)$ and $O(n)$ respectively. The center of the minimum enclosing circle can be reported in $O(\log^2 n)$ time. Very recently, Karmakar et al. [5] proposed an optimal $O(\log n)$ query time algorithm for the query version of the problem. However, the improved query time comes at an increased cost in both preprocessing time and space. The preprocessing time and space complexity for their algorithm is $O(n^2)$.

In this paper we show how to achieve an optimal query time of $O(\log n)$ for the query version of the problem with $O(n \log n)$ preprocessing time and $O(n)$ space. Using our result, we show how to find the smallest enclosing circle where the center is restricted to lie in a set of polygons with a total of m vertices, in $O((m+n) \log n)$ time. This is a significant improvement over the previous best algorithm.

2 Preliminaries

Given a set P of n points in the plane, the only points that can be on the boundary of any enclosing circle lie on the convex hull of P (because a disk is convex and by the definition of the convex hull). Thus, we will assume that the points $P = \{p_1, p_2, \ldots, p_n\}$ are in convex position.

Let $V(P)$ denote the furthest point Voronoi diagram of P (see [8] for a survey on Voronoi diagrams and their furthest point counter-part). The diagram $V(P)$ partitions the plane into n unbounded convex regions, denoted $R(p_1), R(p_2), \ldots, R(p_n)$, such that for any point $p \in R(p_j)$, $d(p, p_j) \geq d(p, p_k)$ for all $k = 1, 2, \ldots, n$, and $k \neq j$. Here, $d(.,.)$ denotes the Euclidean distance between a pair of points. This structure can be computed in $O(n\text{log}n)$ time and $O(n)$ space (see [9]). Furthest point Voronoi diagrams play an important role when studying enclosing disks, because of the following.

[*]Carleton University, Ottawa, Canada, jit@scs.carleton.ca

[†]Chercheur qualifié du FRS-F.N.R.S., Université Libre de Bruxelles, Brussels, Belgium, stefan.langerman@ulb.ac.be

[‡]Tata Consultancy Services Ltd., Pune, India, sasanka.roy@tcs.com

Lemma 1 *[6] The smallest (unconstrained) enclosing circle of a set P of points in the plane always has at least two points of P on its boundary.*

Lemma 1 implies that the center c of the smallest enclosing circle of P always lies on an edge e of $V(P)$. It is this property that allows for the discretization of the problem.

In this paper, for simplicity of exposition, we first assume that the points are in general position (i.e. no four points in P are co-circular). We then show how a minor modification to our solution allows the removal of this general position assumption. We begin by describing how our data structure answers queries when the query object is a line. Then, we show how the algorithm works for query line segments.

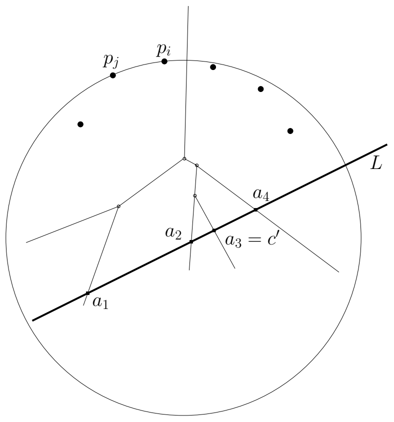

Figure 1: *$\rho(a_i)$ values are unimodal*

It has been shown in [10] how to solve the query problem in $O(\log n)$ time with $O(n \log n)$ preprocessing time and $O(n)$ space when the solution has exactly one point of P on its boundary. If the smallest enclosing circle with center c' constrained to lie on a line segment L has exactly one point of P on its boundary, then c' is either the orthogonal projection of the furthest point to L onto L, or c' must lie on an endpoint of L. Thus, the solution can be found in $O(\log n)$ time by building a point location structure on top of the furthest point Voronoi diagram. The main difficulty is to solve the query problem when the smallest enclosing circle has more than one point of P on its boundary. We present a solution to this problem in the next section.

Lemma 2 *[10] If the smallest enclosing circle with center c' constrained to lie on a segment L has more than one point of P on its boundary, then c' lies on an intersection point of L with an edge of $V(P)$.*

Note that in a degenerate case, c' may be a vertex of $V(P)$ and in this case we can say that c' coincides with the end points of the edge of $V(P)$.

Let $\rho(q)$ denote the radius of the smallest enclosing circle of P with center at point q. Note that if $q \in R(p_i)$ then p_i is on the boundary of the smallest enclosing circle centered at q. Our algorithms will heavily rely on the following:

Lemma 3 *The function $\rho(q)$ is convex.*

Proof. The value of $\rho(q)$ is the maximum distance of q to the points of P, thus $\rho(q)$ is the upper envelope of a set of cones. So it is convex. $\square$

In particular this implies that the restriction of $\rho(q)$ to some line L is a convex function as well.

3 The Data Structure

We now have all the tools to describe our method. Recall our initial assumption that no four points in P are co-circular. Given this assumption, we note that $V(P)$ is a binary tree, denoted T. Let $|T|$ denote the number of vertices of T. Each edge e of T separates two unbounded Voronoi cells. Note that from any point inside an unbounded cell, any ray in an unbounded direction will be entirely contained in the cell. We augment the tree T by associating with each edge such a ray from the midpoint of e for each of the two adjacent cells.

The removal of a Voronoi edge $e = (a, b)$ would split T into two subtrees which we denote by T_a and T_b where T_a is the subtree that contains a and T_b the subtree containing b. The edge e is called a *centroid edge* if T_a and T_b each contain no more than $(2/3)|T|$ vertices. For any binary tree, a centroid edge is known to exist and can be found in linear time [3].

A *centroid decomposition* of T is a binary tree whose nodes are associated with edges of T, whose root is a centroid edge $e = (a, b)$ and whose two subtrees are recursively defined as centroid decompositions of T_a and T_b. It is known that a centroid decomposition of any binary tree T with n vertices has depth $O(\log n)$ and can be constructed in $O(n)$ time [3].

The data structure will be composed of a centroid decomposition of T (see Figure 2) augmented with the rays as described above, and a point location data structure for $V(P)$.

Lemma 4 *The above preprocessing algorithm requires $O(n)$ time and space for a given $V(P)$.*

Lemma 5 *Using the above structure, given a query line L and an edge $e = (a, b)$ of T, we can determine in $O(1)$ time whether the smallest enclosing circle with center constrained to lie on L has its center in e, T_a or T_b.*

Proof. Assume the Voronoi edge $e = (a, b)$ separates the two Voronoi regions $R(p_i)$ and $R(p_j)$. Let ℓ_1 and ℓ_2 be the two rays associated with e. Note that for any

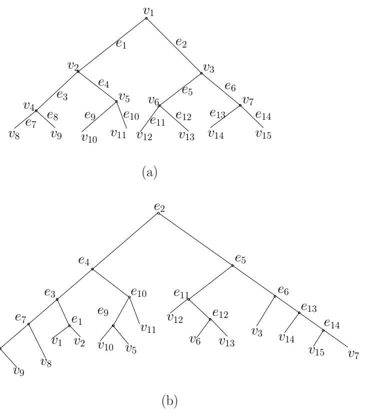

Figure 2: *(a) Binary tree T and (b) Centroid decomposition of T*

point q on ℓ_1 or ℓ_2, we know the cell that contains q, and so which is the furthest point to q. Therefore we can compute the value and the gradient of $\rho(q)$ in $O(1)$ time.

The two rays ℓ_1 and ℓ_2 divide the plane into two regions and the two subtrees T_a and T_b are each wholly contained in one of these regions. Let A be the region containing T_a and B the region containing T_b. The query line L may have three different types of intersections with ℓ_1 and ℓ_2, which form the basis of our case analysis:

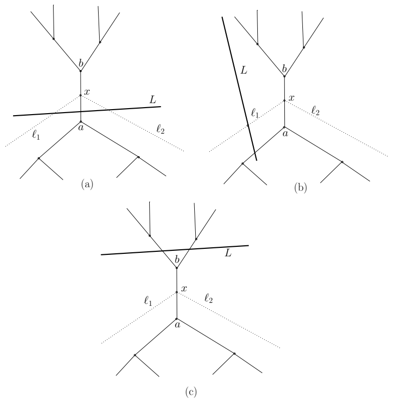

Figure 3: *Illustration of pruning*

Case 1. L intersects both rays ℓ_1 and ℓ_2 (See Figure 3(a))

Case 2. L intersects only one ray, say ℓ_1 (See Figure 3(b))

Case 3. L intersects neither of the two rays (See Figure 3(c))

For each of the three cases, we show how to eliminate one of the subtrees T_a or T_b from the search.

Case 1: Let y and z be the intersection of L with ℓ_1 and ℓ_2, respectively (See Figure 3). By determining the value and the gradient of ρ at y and at z, we can determine if the answer lies on the segment $[yz]$ or if it lies on L outside $[yz]$, since by Lemma 5, we know that the function is convex. If the solution lies on $[yz]$, then the solution lies in $T \setminus T_b$, otherwise it lies on $T \setminus T_a$.

Case 2. Without loss of generality let L intersect ℓ_1 at y (See Figure 3(b)). Again, by finding the value and the gradient of ρ at y, and by Lemma 5, we can determine if the solution lies in A or B. If the solution lies in A then we know it lies in $T \setminus T_b$. Otherwise, it lies in $T \setminus T_a$.

Case 3. If L does not intersect ℓ_1 and ℓ_2 (See Figure 3(c)), then L lies completely inside A or B. We can discard the sub-tree of the region that does not intersect the line L. In the Figure 3(c) the solution lies in $T \setminus T_a$.

$\square$

Lemma 6 *Using the above preprocessed data structure the QSEC problem can be solved in $O(\log n)$ time.*

Proof. In the worst case we may have traverse the worst case depth of centroid decomposition tree which is $O(\log n)$. Each step in this traversal costs $O(1)$ time. Hence the query time complexity result follows. $\square$

4 *QSEC* for Query Line Segment

Here, the query object $L = [f, g]$ is a line segment. We first solve $QSEC$ for the query line $\overline{L}$ that contains the line segment L. Let α be the center of the $QSEC$ for line $\overline{L}$. If α lies inside $[f, g]$, then report α. Otherwise, by Lemma 5, the center c' of the desired constrained smallest enclosing circle is one of the endpoints f or g, which is closest to α. Let f be the closest point of α. Then $\alpha = c'$ and the radius of the desired smallest enclosing circle is $d(p, c')$, where $p \in P$ is the point whose corresponding Voronoi cell contains c'. The Voronoi cell that contains α can be found in $O(\log n)$ time. Thus $QSEC$ for the query line segment can be solved in $O(\log n)$ time.

5 Solution when P is in general position

When P is in general position then T may not be a binary tree. So, Lemma 4 is not true anymore. Now we will describe a method to split the nodes of T whose degree are greater than 3 by adding some virtual edges and construct a virtual binary tree VT. We will show that the number of edges thus inserted is no more than $O(n)$. Then it is easy to observe that solving $QSEC$ problem in VT is same as solving this problem for T and $QSEC$ of VT is $QSEC$ of T. Let us consider a vertex v of T with degree greater than 3. For simplicity, the degree of v is $k > 3$.

We will add $k-1$ virtual edges as follows to make it a binary tree of size $O(k)$:

Let $(e_1, e_2, \ldots, e_k)$ (see Figure 4(a)) be the Voronoi edges adjacent to vertex v. We will insert $k-1$ virtual edges $(ve_1, ve_2, \ldots, ve_{k-1})$ between the pair of edges $(e_1, e_2, \ldots, e_k)$ (see Figure 4(b)). The edge ve_j keep the coordinates of the vertex v and the equation of Voronoi edges e_j and e_{j+1}. Intuitively, the implication of virtual edge ve_j is that we can always draw two rays such that one half-plane, say HP_1, contains the sub-tree that has edges $e_1, e_2, \ldots, e_j$ and has root at v. Other half-plane HP_2 contains the sub-tree that contains the edges $e_{j+1}, e_{j+2}, \ldots, e_k$ and has root at v.

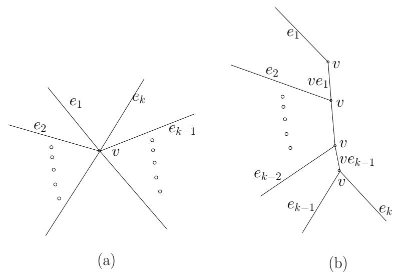

Figure 4: *Splitting a vertex of degree greater than* 3

6 Constrained Smallest Enclosing Circle Problem with Center in a given Set of Polygons

Now for the problem where we have to find the center inside r simple polygons with a total of m edges. Compute the farthest point Voronoi diagram $V(P)$ and identify the center c' of the unconstrained smallest enclosing circle. If it is inside one of these polygons, we report the answer. Otherwise, the center will be on the boundary of one of these polygons. For each edge (line segment), we compute the center of the constrained smallest enclosing circle with center on that edge, and report the radius of the smallest one. Thus, the overall time complexity becomes $O((n+m)\log n)$, where $|P| = n$. Thus it improves the previous running time proposed Roy et al. [10]. There may exist more than one such circle attaining the smallest radius [1]. We can report all these circles with the same time complexity.

References

[1] P. Bose and G. Toussaint, *Computing the constrained euclidean, geodesic and link center of a simple polygon with applications*, Proc. of the Pacific Graphics International, 1996, pp. 102–112.

[2] P. Bose and Q. Wang, *Facility location constrained to a polygonal domain*, Proc. of the Latin American Theoretical Informatics Symposium, 2002, pp. 153–164.

[3] G. N. Frederickson and D.B. Johnson, *Generating and searching sets induced by networks*, Proc. of the 7th International Colloquium on Automata, Languages and Programming, 1980, pp. 221–233.

[4] F. Hurtado, V. Sacristan, and G. Toussaint, *Facility location problems with constraints*, Studies in Locational Analysis **15** (2000), 17–35.

[5] A. Karmakar, S. Roy, and S. Das, *Fast computation of smallest enclosing circle with center on a query line segment*, Proc. of the Canadian conference on Computational Geometry, 2007, pp. 273–276.

[6] D. T. Lee, *Furthest neighbor voronoi diagrams and applications*, Report 80-11-FC-04, Dept. Elect. Engrg. Comput. Sci., Northwestern Univ., Evanston, IL, 1980.

[7] N. Megiddo, *Linear-time algorithms for linear programming in r^3 and related problems*, SIAM Journal on Computing **12** (1983), 759–776.

[8] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial tessellations: Concepts and applications of voronoi diagrams*, 2000.

[9] F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*, 1990.

[10] S. Roy, A. Karmakar, S. Das, and S. C. Nandy, *Constrained minimum enclosing circle with center on a query line segment*, Proc. Mathematical Foundation of Computer Science, 2006, pp. 765–776.

[11] J. J. Sylvester, *A question in the geometry of situation*, Quarterly Journal of Mathematics (1857), 1–79.