Computing the Closest Point to a Circle

Pinaki Mitra
NIMC, Alipore
Calcutta, India
e-mail: pinaki_m@hotmail.com

Asish Mukhopadhyay *
School of Computer Science
University of Windsor, Windsor, Canada
e-mail: asishm@uwindsor.ca

S. V. Rao
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati, India
e-mail: svrao@iitg.ernet.in

Abstract

In this paper we consider the problem of computing the closest point to the boundary of a circle among a set S of n points. We present two algorithms to solve this problem. One algorithm runs in $O(n^3)$ preprocessing time and space and $O(\log^2 n)$ query time. The other algorithm runs in $O(n^{1+\epsilon})$ preprocessing time and $O(n \log n)$ space and $O(n^{2/3+\epsilon})$ query time. Thus we exhibit a trade-off between preprocessing and query times For dimensions $d \geq 3$ we present an algorithm with $O(n^{\lceil d/2 \rceil + \epsilon'})$ preprocessing time to report an approximate closest point to the boundary of d-dimensional query sphere R in $O(n^{1-1/(d+1)+\epsilon})$ query time.

1 Introduction

The problem of preprocessing a set S, of n points in the plane to determine the closest point to a query line was initially addressed by Cole and Yap [2], who obtained a solution with preprocessing time and space in $O(n^2)$ and query time in $O(\log n)$. Lee and Ching [6] obtained the same result using geometric duality. In [8] an algorithm was presented with $O(n \log n)$ preprocessing time and space and $O(n^{0.695})$ query time. The space complexity was improved to $O(n)$ in a subsequent result by Mitra and Chaudhuri [9]. In [10], the simplicial partition technique of [7] was used to improve the query time to $O(n^{1/2+\epsilon})$ for arbitrary $\epsilon > 0$, with preprocessing time and space in $O(n^{1+\epsilon})$ and $O(n \log n)$ respectively.

In this paper, we consider a natural extension of the above problem - determining a point in S that is closest to the boundary of a query circle. Our first algorithm precomputes all higher order Voronoi diagrams. The preprocessing time and space complexities of this algorithm are in $O(n^3)$, while the query time is in $O(\log^2 n)$. We propose another algorithm that uses the simplicial partition idea of Matousek [7]. The preprocessing time and space complexities of this algorithm are in $O(n^{1+\epsilon})$ and $O(n \log n)$ respectively, while the query time is in $O(n^{2/3+\epsilon})$.

The second approach generalizes to higher dimensions. Here, however, point location turns out to be a bottleneck. Recently, some attempts have been made to overcome this problem by computing an approximate nearest neighbor in higher dimensions [4]. Taking a cue from this, for dimension $d \geq 3$, we propose an algorithm that finds an approximate closest point, according to two different approximation schemes, with preprocessing time in $O(n^{\lceil d/2 \rceil + \epsilon'})$ and query time in $O(n^{1-1/(d+1)+\epsilon})$.

2 Geometric Insight for High Preprocessing and Low Query Algorithm

In a preprocessing step, we compute all higher order Voronoi diagrams. The cost of this is in $O(n^3)$ [11]. The main idea of the query procedure is a binary search on the pre-computed higher-order Voronoi diagrams. Let the query circle have centre at p and radius a.

Algorithm Query(R)

1. Compute the nearest neighbor of p from $Vor_1(S)$ and check if its distance from p is less than a. If not, we report the nearest neighbor of p as the answer to the query since in this case all points of S lies outside or on the boundary of the circle R. **Stop**.

2. Compute the furthest neighbor of p from $Vor_n(S)$ and check if its distance from p is greater than a. If not, we report the furthest neighbor as the answer to the query since in this case all points of S lies inside or on the boundary of the circle R. **Stop.**

3. Perform a binary search within the interval $(1 \ldots n)$. Compute $\lfloor n/2 \rfloor$-th nearest neighbor of p from $Vor_{\lfloor n/2 \rfloor}$ and determine its distance from p. If it is equal to a this point is the answer to our query. If it is greater than a recurse in the interval

*Research supported by an NSERC Operating Grant

$(1 \ldots \lfloor n/2 \rfloor)$. Otherwise, if it is less than a recurse in the interval $\lfloor n/2 \rfloor \ldots n$. Thus determine the k-th and $(k+1)$-th nearest neighbors s and t of p such that $(dist(p, s) < a)$ and $(dist(p, t) > a)$.

4. For each of the the two points s and t determined in **Step III**, we compute the absolute difference of their distance from p and a. The point having the least value of the absolute difference is the answer to our query.

Thus we have the following theorem.

Theorem 1 *Given a set S of n points, the closest point from the boundary of a query circle R can be computed in $O(\log^2 n)$ time with $O(n^3)$ preprocessing time and space.*

In the next section, we propose an alternate algorithm based on the simplicial partition technique of Matousek [7] to improve the preprocessing time.

3 Simplicial Partitions

A *simplicial partition* of S is a collection of pairs

$$\Psi(S) = \{(S_1, t_1), (S_2, t_2), \ldots, (S_r, t_r)\} \quad (1)$$

where the S_i's are mutually disjoint subsets of S whose union is S, and each t_i is a tetrahedron that contains S_i (see Fig. 1).

A simplicial partition is *fine* if $|S_i| \leq 2 * n/r$ for each i. For a given simplicial partition $\Psi(S)$, the *crossing number* of a plane Q is the number of tetrahedrons of $\Psi(S)$ that the plane properly intersects. The crossing number of $\Psi(S)$ is the maximum crossing number over all planes that intersect the simplicial partition.

The following theoretically important result by Matousek [7] shows how to construct a fine simplicial partition with a low crossing number.

Theorem 2 *For any given set S of n d-dimensional points and a parameter r, $1 \leq r \leq n$, a fine simplicial partition of size r, with $O(r^{1-1/d})$ crossing number, exists. Further, for any given $\epsilon > 0$, such a simplicial partition can be constructed in time $O(n^{1+\epsilon})$.*

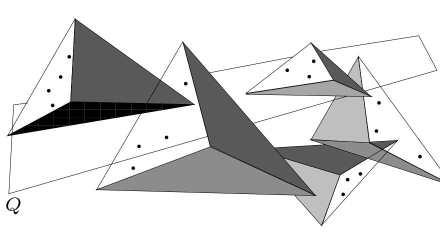

Figure 1: Plane Q cuts a simplicial partition.

We lift the set S of n points and the query circle R, onto the surface of a paraboloid [3] $z = x^2 + y^2$. S' denotes the lifted set of points and R' the plane through the image of the query circle. Points inside the circle R gets mapped to one halfspace and points outside the ring R gets mapped to the other halfspace. We construct a fine simplicial partition of S' of size r with crossing number $O(r^{2/3})$. What is important for our problem is that each tetrahedron that is not intersected by R', traps a set of points whose projections lie completely inside or outside the query circle.

We use the simplicial partition of the lifted set of points to create a r-way partition tree on the set of points S, keeping with each subset of points the corresponding tetrahedron that enclosed their lifted images in 3-space.

4 Algorithmic Details

The constructibility of a fine simplicial partition in 3-space with a low crossing number implies that the projections of the point sets enclosed by at least $r - O(r^{2/3})$ simplices are not intersected by a query circle R. For the remaining at most $O(r^{2/3})$ simplices that are intersected by S' ,we proceed recursively on the respective partitions of the projections of the point sets enclosed by these tetrahedra.

The preprocessing phase of the algorithm maintains both the nearest neighbor and furthest neighbor Voronoi diagrams of the point set corresponding to each node of the partition tree. Then preprocess each of the Voronoi diagram using the algorithm of Kirkpatrick [5] to perform point location.

Given a query circle R with center p and radius a, we answer the closest point query, using the following algorithm.

Algorithm $\mathcal{Q}$

1. Initialize q to (∞, ∞), the nearest point to the the query ring.

2. If the tetrahedron associated with the node of the partition tree lies below R', use the furthest neighbor Voronoi diagram of the projection of the point set inside the tetrahedron to detect the region in which the center p lies. Let t_1 be the point associated with this region. In Fig. 2a, since p lies in $V(p_i)$ the closest point to the boundary of R is p_i and thus $t_1 = p_i$. If the absolute difference of the distance of t_1 from p and a is less than that of q update q to t_1.

3. If the tetrahedron associated with the node of the partition tree lies above R', use the nearest neighbor Voronoi diagram of the projection of the point set inside the quadrilateral to detect the region in

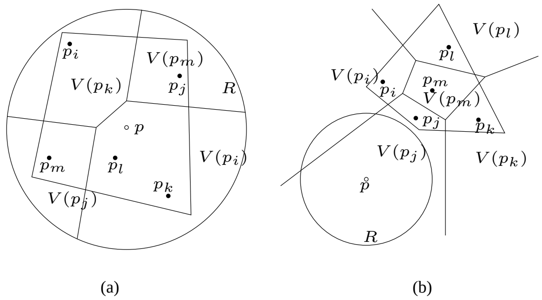

Figure 2: (a) Furthest neighbor Voronoi diagram is queried when the tetrahedron lies below R'. (b) Nearest neighbor Voronoi diagram is queried when the tetrahedron lies above R'.

which the center p lies. Let t_2 be the point associated with this region. In Fig. 2b, since p lies in $V(p_j)$ the closest point to the boundary of R is p_j and thus $t_2 = p_j$. If the absolute difference of the distance of t_2 from p and a is less than that of q update q to t_2.

4. If the tetrahedron associated with the node of the partition tree intersects R' we recurse on the subtrees rooted at that node of the partition tree.

The complexity of the algorithm is summarized in the following theorem.

Theorem 3 *Given a set S of n points and a query circular circle R the closest point from the boundary of R in the set S can be answered in $O(n^{2/3+\epsilon})$ query time with $O(n^{1+\epsilon})$ preprocessing time and $O(n \log n)$ preprocessing space.*

The approach of the previous section extends to higher dimensions in exactly the same way.

5 Approximate nearest point

If an approximate closest point is acceptable, we can avoid point location in higher dimensional Voronoi diagrams required for an exact solution. We preprocess each partitioned point set using Clarkson's [1] randomized algorithm for the nearest neighbor query. For a set S of n points in d-dimension we can preprocess this set in $O(n^{\lceil d/2 \rceil + \epsilon'})$ time to perform nearest neighbor query in $O(\log n)$ time. Now, for the furthest point query in the preprocessing phase we will maintain the furthest point from each point in the simplicial partition. For a partition with m points this computation can be easily done using any brute force algorithm in $O(m^2)$ time in any dimension. Given a query sphere, if the whole partition lies completely inside, we will use the algorithm of Clarkson to compute the nearest neighbor q of the

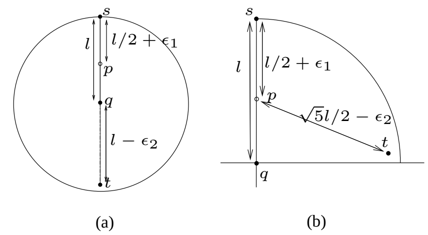

Figure 3: Diagram for the worst case: (a) first approximation (b) second approximation.

center p. Then use this furthest neighbor of q as an approximate furthest neighbor of p.

Fig. 3a illustrates the worst case situation if we adopt this approach. In this diagram we see that the actual furthest neighbor of p, i.e., t is at a distance which is almost 3 times the distance of p from the reported furthest neighbor s.

The constant of approximation can be improved if we increase the preprocessing overhead by maintaining the furthest neighbor of each data point in each of the 2^d quadrants around that point. Again for a partition with m points this computation can be easily done using any brute force algorithm in $O(m^2)$ time in any dimension. Now given the query circle we locate the nearest neighbor q of the center p. Then from $O(2^d)$ furthest points maintained for q we choose the one which is at maximum distance from p. For fixed d thus we have to spend an additional $O(1)$ time besides the $O(\log n)$ time nearest neighbor query. Fig. 3b illustrates the worst case situation if we adopt this approach. In this diagram we see that the actual furthest neighbor of p, i.e., t is at a distance which is almost $\sqrt{5}$ times the distance of p from the reported furthest neighbor s.

Now if we try to bound the error of approximation from the boundary of the query sphere the error ratio is roughly $(a - l/2 - \epsilon_1)/(a - kl/2 + \epsilon_2)$, where $k = 3$ in the first approximation and $k = \sqrt{5}$ in the second approximation. Thus for $a >> l$ we have constant approximation. But for values of a close to $kl/2$ the error of approximation is high.

The time complexity of preprocessing each partition using Clarkson's algorithm [1] would dominate the time complexity of construction of the partition tree. Following a similar analysis as done for two dimension, we establish the following theorem.

Theorem 4 *Given a set of n points and a query sphere R in d dimensions, an approximate nearest neighbor from the boundary of the sphere can be computed in $O(n^{1-1/(d+1)+\epsilon})$ query time with $O(n^{\lceil d/2 \rceil + \epsilon'})$ preprocessing.*

6 Conclusions

In this paper we have shown that the results of Mitra and Chaudhuri [9] can be generalized to answer the circle query problem. We have presented two algorithms one with high preprocessing and low query time and the other one with low preprocessing and high query time.

References

[1] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[2] R. Cole and C. K. Yap. Geometric retrieval problems. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 112–121, 1983.

[3] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 1:25–44, 1986.

[4] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, page to appear, 1998.

[5] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983.

[6] D. T. Lee and Y. T. Ching. The power of geometric duality revisited. *Inform. Process. Lett.*, 21:117–122, 1985.

[7] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[8] P. Mitra. Finding the closest point to a query line. In G. Toussaint, editor, *Snapshots in Computational Geometry*, volume II, pages 53–63. 1992.

[9] P. Mitra and B. B. Chaudhuri. Efficiently computing the closest point to a query line. *Pattern Recognition Letters*, 19:1027–1035, 1998.

[10] Asish Mukhopadhyay. Using simplicial paritions to determine a closest point to a query line. *Pattern Recognition Letters*, 24:1915–1920, 2003.

[11] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, October 1990.