Packing Two Disks into a Polygonal Environment*

Prosenjit Bose[1], Pat Morin[2], and Antoine Vigneron[3]

[1] Carleton University. jit@scs.carleton.ca
[2] McGill University. morin@cs.mcgill.ca
[3] Hong Kong University of Science and Technology. antoine@cs.ust.hk

Abstract. We consider the following problem. Given a polygon P, possibly with holes, and having n vertices, compute a pair of equal radius disks that do not intersect each other, are contained in P, and whose radius is maximized. Our main result is a simple randomized algorithm whose expected running time, on worst case input, is $O(n \log n)$. This is optimal in the algebraic decision tree model of computation.

1 Introduction

Let P be a polygon, possibly with holes, and having n vertices. We consider the following problem, which we call 2-DISK: Find a pair of disks with radius r^* that do not intersect each other, are contained in P, and such that r^* is maximized. Biedl *et al.*[5] gave an $O(n^2)$ time algorithm to solve this problem.

Special cases of 2-DISK have been studied previously. When P is a convex polygon, Bose *et al.* [6] describe a linear time algorithm and Kim and Shin [10] describe an $O(n \log n)$ time algorithm. For simple polygons (i.e. polygons without holes), Bespamyatnikh [4] gives an $O(n \log^3 n)$ time algorithm based on the parametric search paradigm [11].

Another special case occurs when the holes of P degenerate to points. This is known as the *maximin 2-site facility location* problem [3,9]. In this formulation we can think of the centers of the two disks as obnoxious facilities such as smokestacks, or nuclear power plants, and the points as population centers. The goal is maximize the distance between each facility and the nearest population center. Katz *et al.* [9] give an $O(n \log n)$ time algorithm for the decision version of the 2-site facility location problem in which one is given a distance d and asked if there exists a placement of 2 non-intersecting disks of radius d, each contained in P such that no point is included in either of the disks.

In this paper we present a simple randomized algorithm for the general case in which P is not necessarily convex and may contain holes. Our algorithm runs in $O(n \log n)$ expected time. It can also be used to solve the optimization version of the 2-site maximin facility location problem in $O(n \log n)$ time. Finally we

* This research was supported by the Natural Sciences and Engineering Research Council of Canada and by the Hong Kong Research Grant Council CERG grant HKUST6137/98E.

observe that, when we allow polygons with holes, $\Omega(n \log n)$ is a lower bound for 2-DISK by a simple reduction from MAX-GAP.

The remainder of the paper is organized as follows: Section 2 reviews definitions and previous results regarding the medial-axis. Section 3 describes our algorithm. Section 4 summarizes and concludes with an open problem.

2 The Medial-Axis

For the remainder of this paper, P will be a polygon, possibly with holes, and having n vertices. The *medial-axis* $M(P)$ of P is the locus of all points p for which there exists a disk centered at p, contained in P, and which intersects the boundary of P in two or more points. See Fig. 1 for an example. Alternatively, $M(P)$ is a portion of the *Voronoi diagram* of the open line segments and vertices defined by the edges of P. To be more precise, we need to remove the Voronoi edges that are outside P and those associated with an edge and one of its endpoints. It is well known that the medial-axis consists of $O(n)$ straight line segments and parabolic arcs.

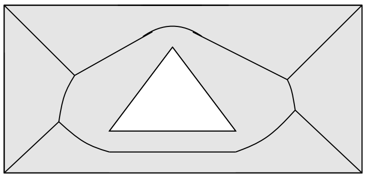

Fig. 1. The medial-axis of a polygon with a triangular hole.

Algorithmically, the medial-axis is well understood. There exists an $O(n)$ time algorithm [7] for computing the medial-axis of a polygon without holes and $O(n \log n)$ time algorithms for computing the medial-axis of a polygon with holes [2]. Furthermore, these algorithms can compute a representation in which each segment or arc is represented as a segment or arc in $\mathbb{R}^3$, where the third dimension gives the radius of the disk that touches two or more points on the boundary of P.

We say that a point $p \in P$ *supports* a disk of radius r if the disk of radius r centered at p is contained in P. We call a vertex, parabolic arc or line segment x of $M(P)$ an *elementary object* if the radius of the largest disk supported by $p \in x$ is monotone as p moves from one endpoint of x to the other. Each edge of $M(P)$ can be split into two elementary objects. Thus, $M(P)$ can be split into $O(n)$ elementary objects whose union is $M(P)$.

3 The Algorithm

Next we describe a randomized algorithm for 2-DISK with $O(n \log n)$ expected running time. We begin by restating 2-DISK as a problem of computing the diameter of a set of elementary objects under a rather unusual distance function. We then use an algorithm based on the work of Clarkson and Shor [8] to solve this problem in the stated time.

The following lemma, of which similar versions appear in Bose *et al.* [6] and Biedl *et al.* [5], tells us that we can restrict our search to disks whose centers lie on $M(P)$.

Lemma 1. *Let D_1 and D_2 be a solution to* 2-DISK *which maximizes the distance between D_1 and D_2 and let p_1 and p_2 be the centers of D_1 and D_2, respectively. Then D_1 and D_2 each intersect the boundary of P in at least two points and hence p_1 and p_2 are points of $M(P)$.*

Proof. Refer to Fig. 2. Suppose that one of the disks, say D_1, intersects the boundary of P in at most one point. Let o_1 be this point, or if D_1 does not intersect the boundary of P at all then let o_1 be any point on the boundary of D_1. Note that there is some value of $\epsilon > 0$ such that D_1 is free to move by a distance of ϵ in either of the two directions perpendicular to the direction $\overrightarrow{p_1 o_1}$ while keeping D_1 in the interior of P. However, movement in at least one of these directions will increase the distance $|p_1 p_2|$, which is a contradiction since this distance was chosen to be maximal over all possible solutions to 2-DISK.

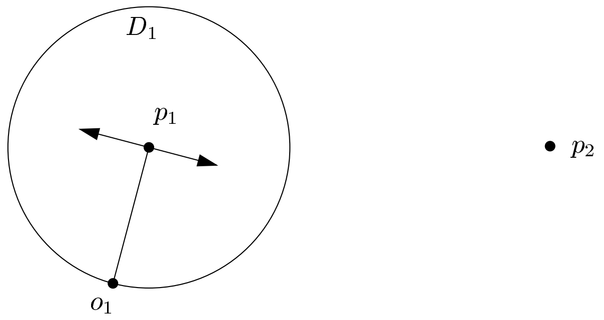

Fig. 2. The proof of Lemma 1

Let x_1 and x_2 be two elementary objects of $M(P)$. We define the *distance* between x_1 and x_2, denoted $d(x_1, x_2)$ as $2r$, where r is the radius of the largest pair of equal-radius non-intersecting disks d_1 and d_2, contained in P and with d_i centered on x_i, for $i = 1, 2$. There are two points to note about this definition of distance: (1) if the distance between two elementary objects is $2r$, then we can place two non-intersecting disks of radius r in P, and (2) the distance from an elementary object to itself is not necessarily 0. Given two elementary objects it is possible, in constant time, to compute the distance between them as well as the locations of 2 disks that produce this distance [5].

Let E be the set of elementary objects obtained by taking the union of the following three sets of elementary objects:

1. the set of vertices of $M(P)$,
2. the set of elementary line segments obtained by splitting each straight line segment of $M(P)$ into at most two elementary objects.
3. the set of elementary parabolic arcs obtained by splitting each parabolic arc of $M(P)$ into at most two elementary objects.

We call the *diameter* of E the maximum distance between any pair $x, y \in E$, where distance is defined as above. Now, it should be clear from Lemma 1 that 2-DISK can be solved by finding a pair of elements in E whose distance is equal to the diameter of E.[1]

Thus, all that remains is to devise an algorithm for finding the diameter of E. Let m denote the cardinality of E and note that, initially, $m = O(n)$. Motivated by Clarkson and Shor [8], we compute the diameter using the following algorithm. We begin by selecting a random element x from E and finding the element $x' \in E$ whose distance from x is maximal, along with the corresponding radius r. This can be done in $O(m)$ time, since each distance computation between two elementary objects can be done in constant time. Note that r is a lower bound on r^*. We use this lower bound to do *trimming* and *pruning* on the elements of E.

We trim each element $y \in E$ by partitioning y into two subarcs,[2] each of which may be empty. The subarc $y_{\geq}$ is the part of y supporting disks of radius greater than or equal to r. The subarc $y_{<}$ is the remainder of y. We then *trim* $y_{<}$ from y by removing y from E and replacing it with $y_{\geq}$. During the trimming step we also remove from E any element that does not support a disk of radius greater than r. Each such trimming operation can be done in constant time, resulting in an $O(m)$ running time for this step.

Next, we prune E. For any arc $y \in E$, the *lowest point* of y is its closest point to the boundary of P. In the case of ties, we take a point which is closest to one of the endpoints of P. By the definition of elementary objects, the lowest point of y is therefore an endpoint of y. The closed disk with radius r centered on the lowest point of y is denoted by $D(y)$. We discard all the elements $y \in E$ such that $D(y) \cap D(x) \neq \emptyset$ for all $x \in E$.

Pruning can be performed in $O(m \log m)$ time by computing, for each lowest endpoint p, a matching lowest endpoint q whose distance from p is maximal and then discarding p if $|pq| \leq 2r$. This computation is known as *all-pairs furthest neighbors* and can be completed in $O(m \log m)$ time [1].

[1] Here we use the term "pair" loosely, since the diameter may be defined by the distance between an elementary object and itself.

[2] We use the term subarc to mean both parts of segments and parts of parabolic arcs.

Once all trimming and pruning is done, we have a new set of elementary objects E' on which we recurse. The recursion completes when $|E'| \leq 2$, at which point we compute the diameter of E' in constant time using a brute-force algorithm. We output the largest pair of equal-radius non-overlapping disks found during any iteration of the algorithm.

To prove that this algorithm is correct we consider a pair of non-intersecting disks D_1 and D_2, each contained in P and having radius r^*, centered at p_1 and p_2, respectively, such that the Euclidean distance $|p_1p_2|$ is maximal. The following lemma shows that p_1 and p_2 are not discarded from consideration until an equally good solution is found.

Lemma 2. *If, during the execution of one round,* $\{p_1, p_2\} \subset \cup E$ *and* $r < r^*$*, then* $\{p_1, p_2\} \subset \cup E'$ *at the end of the round.*

Proof. We need to show that at the end of the round, there exists elementary objects $y_1, y_2 \in E'$ such that $p_1 \in y_1$ and $p_2 \in y_2$. More specifically, we need to show there exists $y_1, y_2 \in E$ such that p_1, respectively p_2 is not trimmed from y_1, respectively y_2, and y_1 and y_2 are not pruned.

To see that p_1 and p_2 are not trimmed from any elementary object that contains them we simply note that p_1 and p_2 both support disks of radius $r^* > r$ and are therefore not trimmed.

To prove that y_1 and y_2 are not pruned we subdivide the plane into two open halfspaces H_1 and H_2 such that all points in H_1 are closer to p_1 than to p_2 and vice-versa. We denote by L the line separating these two halfspaces.

Recall that, after trimming, an elementary object x is only pruned if $D(x) \cap D(y) \neq \emptyset$ for all $y \in E$. We will show that $D(y_1) \subseteq H_1$ and $D(y_2) \subseteq H_2$, therefore $D(y_1) \cap D(y_2) = \emptyset$ and neither y_1 nor y_2 are pruned. It suffices to prove that $D(y_1) \subseteq H_1$ since a symmetric argument shows that $D(y_2) \subseteq H_2$. We consider three separate cases depending on the location of p_1 on $M(P)$.

Case 1: p_1 is a vertex of $M(P)$. In this case we choose y_1 to be the singleton elementary object $\{p_1\}$. Thus, $D(y_1)$ is centered at p_1. Furthermore, the distance between p_1 and L is at least $r^* > r$. Therefore, one point of $D(y_1)$ is contained in H_1 and $D(y_1)$ does not intersect the boundary of H_1 so it must be that $D(y_1) \subseteq H_1$.

Case 2: p_1 lies in the interior of a straight line segment of $M(P)$. Let p_1' be the lower endpoint of y_1. Let θ be the angle (p_2p_1, p_1p_1') (see Fig. 3). If $\theta \in [-\pi/2, \pi/2]$ then we can move p_1 slightly in the direction opposite to p_1' while keeping D_1 inside P, thus contradicting the assumption that $|p_1p_2|$ is maximal. Therefore $\theta \in [\pi/2, 3\pi/2]$, which implies that $D(y_1)$ lies in H_1.

Case 3: p_1 lies in the interior of a parabolic arc of $M(P)$. In this case D_1 is tangent to an edge e_1 of P and touches one of its vertices v. p_1' still denotes the lower endpoint of y_1. Without loss of generality, assume that e_1 is parallel to the x–axis and $x(p_1') < x(p_1)$ (see Fig. 4). Let L' be the line parallel to L that crosses the segment $[p_1, p_2]$ and that is tangent to D_1. We denote by o_1 the point where e_1 is tangent to D_1, and we denote by o_1' the point such that (o_1, o_1') is a diameter of D_1. Finally, the convex hull of D_1 and $D(y_1)$ is denoted by $\mathcal{C}$.

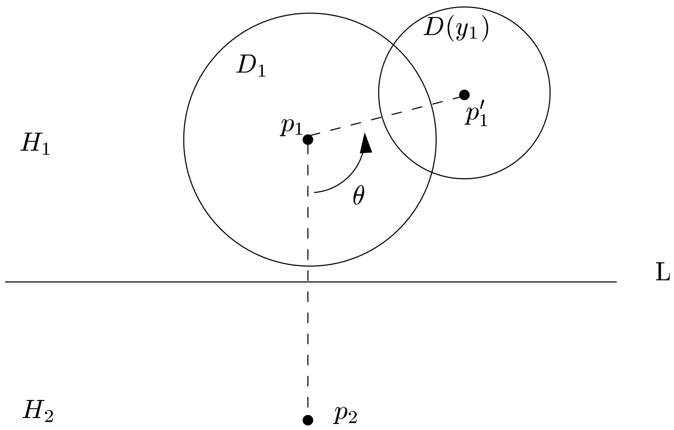

Fig. 3. The proof of Lemma 2 Case 2

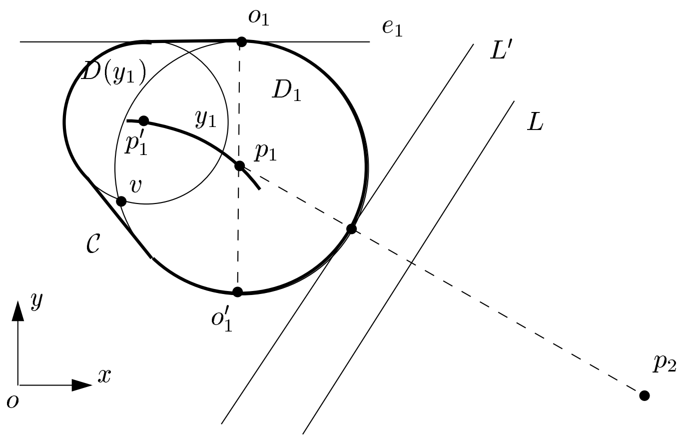

Fig. 4. The proof of Lemma 2 Case 3.

It must be that $\mathrm{x}(p_2) > \mathrm{x}(p_1)$, otherwise p_1 and D_1 could be moved in the positive x direction while keeping D_1 in P. This would increase the distance $|p_1 p_2|$ which is defined as maximal. It follows that L' is tangent to D_1 along the counterclockwise arc (o_1', o_1). Then L' is tangent to $\mathcal{C}$, so by convexity $\mathcal{C}$ lies on the same side of L' as p_1 which implies that D_1 is contained in H_1.

Let d_i denote the distance of the furthest element in E from x_i, and suppose for the sake of analysis that the elements of E are labeled $x_1, \ldots, x_n$ so that $d_i \leq d_{i+1}$. The following lemma helps to establish the running time of the algorithm.

Lemma 3. *If we select $x = x_i$ as the random element, then we discard all $x_j \in E$ such that $j \leq i$ from E.*

Proof. For any $j \leq i$, either x_j does not support a disk of radius greater than d_i, or every point on x_j that supports a disk of radius d_i is of distance at most d_i from any other point of $M(P)$ that supports a disk of radius d_i.

In the first case, x_j is removed from E by trimming. In the second case, $D(x_j) \cap D(x_k) \neq \emptyset$ for all $x_k \in E$ and x_j is removed by pruning.

Finally, we state and prove our main theorem.

Theorem 1. *The above algorithm solves* 2-DISK *in $O(n \log n)$ expected time.*

Proof. The algorithm is correct because, by Lemma 2, it never discards p_1 nor p_2 until it has found a solution with $r = r^*$, at which point it has already found an optimal solution that will be reported when the algorithm terminates.

To prove the running time of the algorithm, we use the following facts. Each round of the algorithm can be completed in $O(m \log m)$ time where m is the cardinality of E at the beginning of the round. By Lemma 3, when we select x_i as our random element, all elements x_j with $j \leq i$ disappear from E. Therefore, the expected running time of the algorithm is given by the recurrence

$$T(m) \leq \frac{1}{m} \sum_{i=1}^{m} T(m-i) + O(m \log m) \ ,$$

which readily solves to $O(m \log m)$. Since $m \in O(n)$, this completes the proof.

4 Conclusions

We have given a randomized algorithm for 2-DISK that runs in $O(n \log n)$ expected time. The algorithm is considerably simpler than the $O(n \log^3 n)$ algorithm of Bespamyathnikh [4] and has the additional advantage of solving the more general problem of polygons with holes. Although we have described our algorithm as performing computations with distances, these can be replaced with squared distances to yield an algorithm that uses only algebraic computations.

In the algebraic decision tree model of computation, one can also prove an $\Omega(n \log n)$ lower bound on any algorithm for 2-DISK through a reduction from MAX-GAP [12]. Suppose that the input to MAX-GAP is $y_1, \ldots, y_n$. Without loss of generality one can assume that $y_1 = \min\{y_i : 1 \leq i \leq n\}$ and $y_n = \max\{y_i : 1 \leq i \leq n\}$. We then construct a rectangle with top and bottom sides at y_1 and y_n, respectively, and with width $2(y_n - y_1)$. The interior of this rectangle is then partitioned into rectangles with horizontal line segments having y coordinates $y_1, \ldots, y_n$. See Fig. 5 for an example.

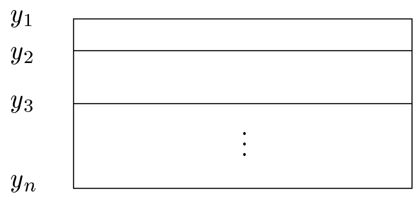

Fig. 5. Reducing MAX-GAP to 2-DISK.

It should then be clear that the solution to 2-DISK for this problem corresponds to placing two disks in the rectangle corresponding to the gap between y_i and y_{i+1} which is maximal, i.e., it gives a solution to the original MAX-GAP problem. Since this reduction can be easily accomplished in linear time and MAX-GAP has an $\Omega(n \log n)$ lower bound, this yields an $\Omega(n \log n)$ lower bound on 2-DISK.

The above reduction only works because we allow polygons with holes. An interesting open problem is that of determining the complexity of 2-DISK when restricted to simple polygons. Is there a linear time algorithm?

References

1. P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees, and related problems in higher dimensions. *Comput. Geom.: Theory & Appl.*, 4:189–201, 1992.
2. Helmut Alt and Otfried Schwarzkopf. The Voronoi diagram of curved objects. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 89–97, 1995.
3. Boaz Ben-Moshe, Matthew J. Katz, and Michael Segal. Obnoxious facility location: Complete service with minimal harm. In *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99)*, pages 76–79, 1999.
4. S. Bespamyatnikh. Draft: Efficient algorithm for finding two largest empty circles. In *Proceedings of the 15th European Workshop on Computational Geometry (EuroCG'99)*, pages 37–38, 1999.
5. T. C. Biedl, E. D. Demaine, M. L. Demaine, A. Lubiw, and G. T. Toussaint. Hiding disks in folded polygons. In *Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG'98)*, 1998.

6. P. Bose, J. Czyzowicz, E. Kranakis, and A. Maheshwari. Algorithms for packing two circles in a convex polygon. In *Proceedings of Japan Conference on Discrete and Computational Geometry (JCDCG '98)*, pages 93–103, 1998.
7. F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete and Computational Geometry*, 21, 1999.
8. K. L. Clarkson and P. W. Shor. Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (SoCG'88)*, pages 12–17, 1988.
9. Matthew J. Katz, Klara Kedem, and Michael Segal. Improved algorithms for placing undesirable facilities. In *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99)*, pages 65–67, 1999.
10. S. K. Kim and C.-S. Shin. Placing two disks in a convex polygon. *Information Processing Letters*, 73, 2000.
11. N. Megiddo. Applying parallel computation algorithms to the design of serial algorithms. *Journal of the ACM*, 30:852–865, 1983.
12. F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985.