Placing Two Disks in a Convex Polygon

Sung Kwon Kim* Chan-Su Shin† Tae-Cheon Yang‡

Keywords: Computational geometry, optimization, convex polygon.

1 Introduction

Let P be a convex polygon of n vertices. We consider a problem (and some of its variants) of finding a pair of largest equiradial non-overlapping disks in P. Regarding P as a sheet of paper, one can fold P along the bisector of the line segment connecting the centers of the disks. Then those two disks coincide each other in the folded polygon. This means the problem is identical to that of finding a largest disk which can be hidden with P by one straight fold. Both sides of the hidden disk should be covered by P. For an illustration, refer to Figure 1 (a). This problem is well motivated from the gift wrapping problem that determine whether a gift is wrapped up (or hidden) using a given paper through one-straight fold.

Consider a line l passing through two points on two different edges of P. The line l divides P into two convex subpolygons $P_1(l)$ and $P_2(l)$. We want to find a *folding line* l which gives two largest equiradial non-overlapping disks, one in $P_1(l)$ and the other in $P_2(l)$. If we fold P along l, then the disks coincide in the folded polygon. A disk of the same radius can be hidden in P. We then define two problems according to the type of the folding line as follows:

P1: Find an optimal folding line for a convex polygon P. Note that a folding line in this problem does not necessarily pass through vertices of P.

P2: Find an optimal folding line for a convex polygon P so that it passes through two vertices of P.

Problem **P1** was investigated in [4, 3]. Biedl et al. [4] considered the problem for a simple polygon, and presented a polynomial-time algorithm; for a convex polygon, their algorithm runs in $O(n^2)$ time. Recently, Bespamyatnikh [3] reduced the time bound to $O(n \log^3 n)$ by using parametric search technique [7]. In this paper, we present an algorithm for a convex polygon whose running time is $O(n \log n)$. Our algorithm does not employ parametric search technique which is known to be hard to implement [1].

For problem **P2**, to our knowledge, no algorithm has been known so far. We solve **P2** in $O(n^2 \log^2 n)$ time by adapting parametric search technique.

*Dept. of Computer Science and Engineering, Chung-Ang Univ., Korea. skkim@cau.ac.kr
†Dept. of Computer Science, HKUST, Hong Kong. cssin@cs.ust.hk
‡School of Information Science, Kyungsung Univ., Korea. tcyang@csd.kyungsung.ac.kr

Let us now modify these problems in a slightly different way. As above, define two subpolygons $P_1(l)$ and $P_2(l)$ of P by a line l intersecting P. Let $r_i(l)$ be the radius of the largest disk inscribed in $P_i(l)$ for $i = 1, 2$. Our new problems are to find a *separating line* l that gives

$$\max_l(\min(r_1(l), r_2(l))).$$

Two different problems are possible according to the type of a separating line:

Q1: Find an optimal separating line when the maximum in the above equation is taken over all lines intersecting P.

Q2: Find an optimal separating line when the maximum in the above equation is taken over all lines passing through two vertices of P.

As will be seen later, problem **Q1** is identical to problem **P1**, so we can solve **Q1** in $O(n \log n)$ time. Hence, the problem of finding two largest non-overlapping disks in a convex polygon is equivalent to problems **P1** and **Q1**. But problem **Q2** is entirely different from problem **P2**. We will prove that **Q2** can be solved in $O(n \log^2 n)$ time by a divide-and-conquer approach.

Besides, there is a related problem of finding two largest disks whose union covers (the boundary of) a convex polygon, instead of inscribing two disks in a convex polygon. The problem was investigated, together with some variants, in [9].

Throughout this paper, we denote by P a convex polygon of n vertices. As the general-position assumption in other geometric algorithms, we assume here that no four or more edges of P touch a common disk. Here, "touch" means just "contact" (not pierce). In addition, we assume that there are no parallel edges in P. These assumptions about input polygons can be easily removed by several techniques [5].

2 Algorithm for problem P1

In this section, we will consider the problem **P1**, which is to find a largest disk hidden in P by one straight fold. As stated in the previous section, this is equivalent to finding two largest equiradial non-overlapping disks in P, so in what follows, we will explain how to find those disks in P.

Let D_1 and D_2 be two optimal equiradial disks in P and let $r^\star$ be their radius. We shall denote the boundary of some closed region R by ∂R. Let us begin with stating two simple, but crucial, lemmas.

Lemma 1 *D_1 and D_2 touch each other; and each of D_1 and D_2 touches (at least) two edges of P.*

Proof: If D_1 and D_2 do not touch each other, we can immediately enlarge both disks to obtain a pair of larger disks. So, D_1 and D_2 must touch each other. If D_1, touching D_2, touches none of the edges of P, then D_1 can be moved away from D_2 so that they no longer touch. If D_1, touching D_2, touches only one edge of P, then D_1 can

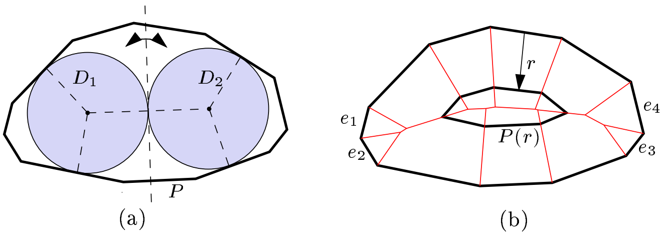

Figure 1: (a) A configuration of two optimal disks D_1 and D_2. (b) A shrunken polygon $P(r)$ of P. The thin lines represent the *medial axis* of P. Note that some edges of P such as e_1, e_2, e_3, e_4 disappear in $P(r)$.

still be moved so that it no longer touches D_2. □

From the above lemma, D_i has (at least) three touching points: (at least) two edge-touching points and one disk-touching point. Note that D_i may have (at most) three edge-touching points by the general-position assumption. However, assume, for a simplicity, that D_i touches exactly two edges and the other disk (refer to Figure 1 (a)). Since such three touching points fix each disk immovable in P, they must form an acute-triangle of all inner angles $< \pi/2$.

Lemma 2 *Two edge-touching points and one disk-touching point of D_i form an acute-triangle containing the center of D_i for $i = 1, 2$.*

Let us now describe an $O(n \log n)$-time algorithm for problem **P1**. The locations at which the centers of D_i's may lie are closely related to the *medial axis* [8] of P. The medial axis of P is defined as the locus of all centers of disks contained in P that touch ∂P, at two or three points (see Figure 1 (b)). In other words, the medial axis of P is a skeleton of the Voronoi diagram for the edges of P, excluding their end vertices. Thus, the medial axis consists of Voronoi vertices and Voronoi edges (which are straight line segments). It can be computed in linear time [2].

From Lemma 1, we know that the centers of D_i's lie on some edges of the medial axis of P. A straightforward way to solve **P1** is to compute two largest equiradial disks decided by each pair of edges of the medial axis of P. The maximum among the computed radii is what we want. It takes $\Omega(n^2)$ time in total. As will be seen later, we do not need to consider all such $O(n^2)$ pairs; only $O(n)$ pairs are enough.

Before explaining it, we will first consider the decision version of problem **P1** as follows: Given a fixed radius $r > 0$, decide whether two non-overlapping disks of radius r can be placed in P. Let $P(r)$ be a *shrunken polygon* of P that is a locus of points in P at distance r from ∂P (see Figure 1 (b)). If a disk of radius r has its center on any location of $\partial P(r)$, then the disk is entirely contained in P. The shrunken polygon $P(r)$ can be easily constructed in linear time by walking around ∂P [6, 10]. Note that some edges of P may disappear in $\partial P(r)$. Next, compute a farthest vertex pair (i.e., diameter pair) of $P(r)$ in linear time [8]. If the diameter is greater than or equal to $2r$, then we can place two non-overlapping disks of radius r

with their centers at the farthest vertex pair of $P(r)$. Otherwise, we have no feasible placements.

Lemma 3 *Given a fixed radius $r > 0$, one can decide in linear time whether two non-overlapping disks of radius r can be placed in P, i.e., whether $r \leq r^\star$, where $r^\star$ is the radius of two largest equiradial non-overlapping disks in P.*

Let M be the medial axis of P. Consider a vertex ν of M. Then we can draw a disk of some radius r, centered at ν and touching three edges of P that define ν in M. Collect all such radii associated with the vertices of M, and sort them in non-decreasing order, $r_1, r_2, \cdots, r_m$. Note that $m = O(n)$, and the disk of radius r_m is the largest disk inscribed in P. Clearly, there is an index j ($1 \leq j < m$) such that $r_j \leq r^\star < r_{j+1}$. We can find j in $O(n \log n)$ time by performing a binary search on the sorted list of r_i's with the help of our decision algorithm (see Lemma 3).

Shrink P by r_j to get $P(r_j)$. Some edges of P may disappear in $\partial P(r_j)$. Expand $P(r_j)$ by r_j outwardly and denote by P' the expanded polygon. Refer to Figure 2 (b). Since P' is obtained by simply scaling $P(r_j)$ by r_j, every edge of $P(r_j)$ appears in $\partial P'$. However, some edges of P may not appear in $\partial P'$, and thus $P \subseteq P'$.

Lemma 4 *Let r' be the radius of two largest non-overlapping disks in P'. Then, $r' = r^\star$.*

Proof: It clearly holds that $r^\star \leq r'$ because $P \subseteq P'$. We assume that $r^\star < r'$, and will show that this leads to a contradiction.

Let $P(r')$ and $P'(r')$ be the shrunken polygons of P and P' by r', respectively. We show that $P(r')$ is actually identical to $P'(r')$, i.e., $P(r') = P'(r')$. If they are identical, then we can place two disks of radius r' in P, which contradicts to the optimality of $r^\star$ for P. Thus, we can conclude that $r^\star = r'$.

Since $P(r') \subseteq P'(r')$ clearly, we assume that $P(r') \subset P'(r')$. This implies there must be an edge e_1 of $P(r')$ such that no edges of $P'(r')$ contain e_1 entirely. Let e be the original edge of e_1 in P, that is, e is shrunken to e_1 in $P(r')$. Let e_2 be the shrunken edge of e in $P(r_j)$. In fact, e_2 does not exist. If e_2 exists in $P(r_j)$, then e_2 would be expanded to some edge e_3 of P'. Remind that P' is an expanded polygon of $P(r_j)$ by r_j, and thus $e \subseteq e_3$. The edge e_3 is again shrunken to an edge e_4 in $P'(r')$. Finally, $e_1 \subseteq e_4$, which contradicts to the fact that no edges in $P'(r')$ contain entirely e_1. Thus e_2 does not exist in $P(r_j)$. Note that e is shrunken to an empty edge in $P(r_j)$, while, to (a non-empty edge) e_1 in $P(r')$ for $r' > r_j$. This is impossible, so $P(r') = P'(r')$. $\Box$

Lemma 4 means that it is enough to compute two largest disks in P', instead in P. In what follows, we will show P' has a nice property to compute the disks and present in detail how to compute them. Consider two shrunken polygons $P(r_j)$ and $P(r_{j+1})$. Let M' be the part of M belonging to $P(r_j) \backslash P(r_{j+1})$, that is, the clipped part of M by $P(r_j) \backslash P(r_{j+1})$ (see Figure 2 (a)). Note that the region $P(r_j) \backslash P(r_{j+1})$ includes $\partial P(r_j)$, but does not include $\partial P(r_{j+1})$. Since r_j is the maximum among all

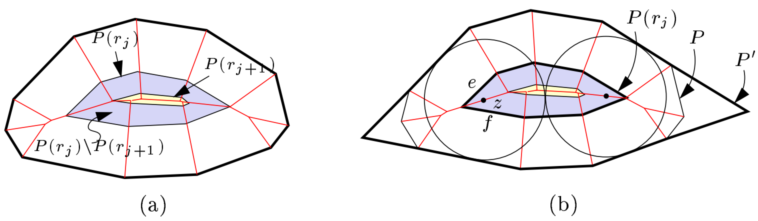

Figure 2: (a) A shaded region represents $P(r_j)\backslash P(r_{j+1})$. (b) A convex polygon P, a shrunken polygon $P(r_j)$, and an expanded polygon P'.

r_i's $\leq r^\star$, there are no vertices of M inside $P(r_j)\backslash P(r_{j+1})$ (possibly, some vertices of M may be on $\partial P(r_j)$). In other words, M' consists of only pieces of some edges of M clipped by $P(r_j)\backslash P(r_{j+1})$. We want to notice here that M' is actually a subset of a medial axis for $P(r_j)$. Since $r_j \leq r^\star < r_{j+1}$, the optimal disks, D_1 and D_2, for P' have their centers on some edges of M'. Suppose that one of the disks, say D_1, has its center on an edge z of M'. As shown in Figure 2 (b), since z is a medial axis edge for $P(r_j)$, it is defined by two edges e, f of $\partial P(r_j)$. The edges e and f have to be adjacent on $\partial P(r_j)$. Remind that P' is an expanded polygon of $P(r_j)$ by r_j. So, the expanded edges of e and f are also adjacent on $\partial P'$. From this observation, we have the following lemma.

Lemma 5 *For each optimal disk for P', two edges of $\partial P'$ at which the disk touches are adjacent on $\partial P'$.*

The remaining is to compute, for each pair of vertices of P', two largest disks determined by its incident four edges. For one vertex pair, we can easily compute the optimal disks in $O(1)$ time [4], so we can consider all possible $O(n^2)$ pairs in $O(n^2)$ time. But, the following lemma states it is sufficient to consider only $O(n)$ *antipodal pairs* of vertices of P'. A pair of vertices is said to be *antipodal* if it admits parallel supporting lines [8]. All antipodal pairs of a convex polygon are found in linear time [8].

Lemma 6 *To compute two largest disks for P', we need consider only $O(n)$ antipodal pairs of vertices of P'.*

Proof: Consider a line l that connects two centers of the optimal disks D_1 and D_2 for P'. The line l passes through the point at which D_1 and D_2 touch each other. By Lemma 2, two edge-touching points of each D_i must be located in the opposite sides with respect to l. This means one can draw two parallel lines supporting a pair of vertices that are incident to touching-edges; such parallel lines will be perpendicular to l. $\square$

The above lemma allows us to finally compute two largest disks for P by checking only $O(n)$ antipodal pairs of vertices of P' in $O(n)$ time. Since we need $O(n \log n)$ time to sort radii r_i's and find r_j, our algorithm runs in $O(n \log n)$ time in total.

Theorem 1 *Given a convex polygon P of n vertices, we can find a largest disk in $O(n \log n)$ time that can be hidden in P when P is folded along a line intersecting P.*

3 Algorithm for problem P2

Let us now consider problem **P2** of finding a largest disk that can be hidden in a convex polygon P when we fold P along a line passing through two vertices of P. This problem is solved in $O(n^2 \log^2 n)$ by simply adapting a parametric search technique [7].

Parametric search is an optimization technique which can be applied in situations where we seek a maximum parameter $r^\star$ satisfying the *monotone* condition that is met by all $r \leq r^\star$ but not by any $r > r^\star$. The strategy of the parametric search is to design efficient sequential and parallel algorithms for the corresponding decision problem: decide whether a given parameter r is smaller than or larger than or equal to the maximum parameter $r^\star$.

Problem **P2** clearly satisfies the monotone condition that one can hide a disk of all radius $r \leq r^\star$ in P, but not for any radius $r > r^\star$. The decision problem for **P2** is as follows: Given a radius $r > 0$, decide whether a disk of radius r can be hidden in P when one folds P along a line passing through two vertices of P. Refer to Figure 3 (a).

Assume that we have a sequential algorithm A_s which runs in $O(T_s)$ time, and a parallel algorithm A_p which runs in $O(T_p)$ time using W_p processors. Then the optimization problem is solved in $O(W_p T_p + T_s T_p \log W_p)$ time; see [7] for details. In this section, we will present a sequential algorithm with $T_s = O(n^2)$ and a parallel algorithm with $T_p = O(\log n)$ and $W_p = O(n^2)$, so **P2** is solved in $O(n^2 \log^2 n)$ time.

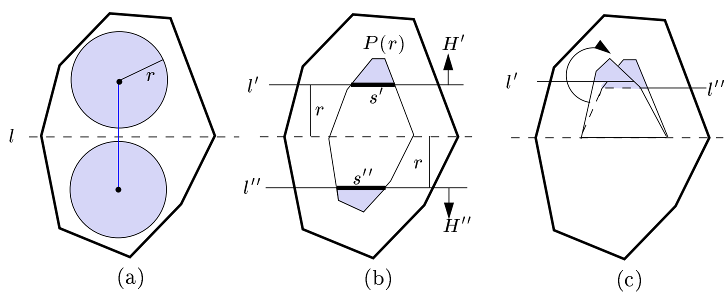

Figure 3: (a) Two disks of radius r are contained in P, i.e., the answer of the decision problem is "yes". (b) Shaded regions are A' and A''. (c) Folding $P(r)$ along l.

Let l be a line through two vertices of P. For simplicity, regard l as the x-axis. Given a value of $r > 0$, we will check whether a disk of radius r can be hidden when P is folded along l. Let l' and l'' be the lines apart from l by distance r in $+y$ and $-y$ directions, respectively. Let H' (resp., H'') be the halfplane bounded below by l' (resp., above by l''). See Figure 3 (b). Consider a shrunken polygon $P(r)$ of P.

Define $A' = P(r) \cap H'$ and $A'' = P(r) \cap H''$. A' (resp., A'') is the region of points at which a disk of radius r can center so that the disk is contained in the upper (resp., lower) subpolygon of P divided by l. We fold $P(r)$ along l as shown in Figure 3 (c). In the folded polygon, if $A' \cap A'' \neq \emptyset$, then we can hide a disk of radius r in P by putting its center at a point in $A' \cap A''$. Otherwise, no disk of radius r can be hidden in P.

Hence, we need to check whether $A' \cap A''$ in the folded polygon of $P(r)$ is empty. One simple way is to compute their actual intersection. However this needs time at least in linear to the number of vertices involved, possibly $\Omega(n)$. Define two line segments, $s' = P(r) \cap l'$ and $s'' = P(r) \cap l''$. Then the following simple lemma allows us to reduce the checking time down to $O(1)$.

Lemma 7 *In the folded polygon of $P(r)$, $A' \cap A'' \neq \emptyset$ if and only if $s' \cap s'' \neq \emptyset$.*

Proof: If $s' \cap s'' \neq \emptyset$, then clearly $A' \cap A'' \neq \emptyset$. For a contradiction, assume that $s' \cap s'' = \emptyset$, but $A' \cap A'' \neq \emptyset$. Let q be a point in $A' \cap A''$. In $P(r)$, q defines two "twin" points, one in A' and the other inA''. Let λ be the vertical line segment connecting these twin points. Since $P(r)$ is convex, $\lambda \in P(r)$. So, in $P(r)$, λ intersects s' and s'' at t' and t'', respectively. Points t' and t'' are twin points as they coincide at a point, t, in the folded polygon of $P(r)$. This implies $t \in s' \cap s''$, which is a contradiction. $\Box$

Let us summarize our decision algorithm. First, compute the shrunken polygon $P(r)$. For each vertex v of P, do the following steps. (i) For each line l passing through v and v's non-adjacent vertices of P, compute s' and s'' and check if they intersect when folding $P(r)$ along l. (ii) If there is a pair of vertices such that s' and s'' intersect, then the answer is "yes". If there is no such pair, the answer is "no".

Let us analyze the time complexity of a sequential implementation of the algorithm. Clearly, the total running time depends on the step (i). For a fixed vertex v, we can compute all s' and s'' in linear time by traversing edges of $P(r)$ one by one like the method [8] of finding antipodal pairs of vertices in a convex polygon. This is possible because both P and $P(r)$ are convex. Since we have to perform step (i) for all n vertices of P, the total time becomes $O(n^2)$, i.e., $T_s = O(n^2)$.

A parallel implementation is more straightforward. Computing $P(r)$ is done in $O(\log n)$ time with $O(n)$ processors as follows: Assign a processor to each medial-axis edge and compute a point on it in constant time whose distance to ∂P is exactly r. Next, angular-sort the points with respect to the center of the largest inscribing circle of P in $O(\log n)$ time with $O(n)$ processors. To obtain $P(r)$, just connect the points in the sorted order. For the remaining step, assign a processor to each pair of vertices of P and compute their corresponding s' and s'' in $O(\log n)$ time through a binary search on $\partial P(r)$. As a result, $T_p = O(\log n)$ and $W_p = O(n^2)$. Plugging these bounds into the parametric search engine, we have the following result.

Theorem 2 *Given a convex polygon P of n vertices, we can find a largest disk in $O(n^2 \log^2 n)$ time that can be hidden in P when P is folded along a line passing through two vertices of P.*

4 Algorithms for problems Q1 and Q2

In this section, we will consider problems **Q1** and **Q2**. Let $r_i(l)$ be the radius of the largest disk inscribed in $P_i(l)$, where $P_i(l)$ for $i = 1, 2$ is a subpolygon of P divided by a separating line l intersecting P.

Problem **Q1** is to find a separating line that maximizes the value $\min(r_1(l), r_2(l))$ for all lines l intersecting P. We first show that problem **Q1** is actually identical to problem **P1**. An obvious fact is that the optimal two disks of problem **Q1** must be equiradial; otherwise, one can shrink the larger one and enlarge the smaller one to increase the value of $\min(r_1(l), r_2(l))$. Another fact is that the two optimal disks touch each other on the separating line. Its proof is quite similar to that of Lemma 1. These facts directly lead that an optimal separating line in **Q1** is just the perpendicular bisector of the line segment connecting the centers of optimal two disks in **P1**. As a result, problem **Q1** can be solved in $O(n \log n)$ time by Theorem 1. In what follows, we will consider only problem **Q2**.

Problem **Q2** is to find a separating line that maximizes the value $\min(r_1(l), r_2(l))$ for all lines l passing through two vertices of P. To solve it, we shall apply a divide-and-conquer technique. Fix a vertex v of P and number the vertices of P counterclockwise $v_0 (= v), v_1, \cdots, v_{n-1}$. Denote by l_j a line connecting v and (its non-adjacent vertex) v_j for $2 \leq j \leq n-2$. Then l_j partitions P into two subpolygons $P_1(l_j)$ and $P_2(l_j)$; $P_1(l_j)$ consists of vertices $v_0, v_1, \cdots, v_j$ and $P_2(l_j)$ does of $v_j, v_{j+1}, \cdots v_{n-1}, v_0$. As the index j increases, $r_1(l_j)$ monotonically increases and $r_2(l_j)$ monotonically decreases. From this monotonicity, we can easily prove that there are at most two local maxima among the values of $\min(r_1(l_j), r_2(l_j))$'s, and moreover, they must be consecutive. For simplicity of the algorithm description, assume that we have only one maxima at the line l_k for some $2 \leq k \leq n-2$. (For the case that two local maxima exist, a similar argument is used.)

Lemma 8 *Let l be a line passing through any pair of vertices u and w, where $u, w \in \{v_1, v_2, \cdots, v_{k-1}\}$ or $u, w \in \{v_{k+1}, v_{k+2}, \cdots, v_{n-1}\}$. Then*

$$\min(r_1(l), r_2(l)) < \min(r_1(l_k), r_2(l_k)).$$

Proof: Assume that $u = v_i$ and $w = v_j$, where $1 \leq i < j \leq k-1$. Then the counterclockwise ordering on ∂P is $v_0 = v$, v_i, v_j, and v_k. For a contradiction, suppose that the inequality does not hold for the pair (v_i, v_j). Consider a line l_j through v and v_j. Then, from the monotonicity, $\min(r_1(l_j), r_2(l_j))$ is strictly larger than $\min(r_1(l_k), r_2(l_k))$. This means the line l_j is a new maximum for v, which contradicts to the optimality of l_k. □

From this lemma, we need to consider only pairs (u, w) of vertices with $u \in \{v_0, v_1, \cdots, v_k\}$ and $w \in \{v_k, v_{k+1}, \cdots, v_{n-1}, v_0\}$. Consequently, the number of pairs to be considered is reduced by almost half. We now apply a divide-and-conquer technique with two vertex chains, $S_1 = \{v_0, v_1, \cdots, v_k\}$ and $S_2 = \{v_k, \cdots, v_{n-1}, v_0\}$.

The recursive part is summarized as follows: (i) Pick the middle vertex, $k/2$-th one, from S_1 and denote it by v. (ii) Find a vertex v' in S_2 that gives the local maximum for v. Note that l_k is a line passing through v and v'. Then v divides S_1 into two chains of equal size, S_{11} and S_{12}, in the counterclockwise order. Also, v' divides S_2 into two chains, S_{21} and S_{22}, in the counterclockwise order. By Lemma 8, we need to consider only vertex-pairs between S_{11} and S_{21}, and between S_{12} and S_{22}. As a result, we further eliminate a half of vertex-pairs. (iii) Perform steps (i)-(ii) recursively with $S_1 = S_{11}$ and $S_2 = S_{21}$, and with $S_1 = S_{12}$ and $S_2 = S_{22}$. Recursion stops when S_1 consists of only one vertex; at that time, the local maximum for the vertex is found in S_2 as in (ii).

Let us analyze time complexity. Since a largest inscribed disk of a convex polygon can be found in linear time, the value of $\min(r_1(l), r_2(l))$ can be computed in linear time by finding largest disks in $P_1(l)$ and $P_2(l)$. The values of $\min(r_1(l_j), r_2(l_j))$ have at most two (consecutive) local maxima, so we can perform a binary search to locate l_k. Step (ii) thus takes $O(n \log n)$ time. The recursion will stop after $O(\log n)$ times of recursive calls. Hence, the total running time is $O(n \log^2 n)$.

Theorem 3 *Given a convex polygon P of n vertices, we can find the maximum value of the minimum radius of two disks of P in $O(n \log^2 n)$ time if the separating lines are restricted to ones through two vertices of P.*

References

[1] P.K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. Tech. Report CS-1996-19, Dept. Computer Science, Duke University, 1996.

[2] A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989.

[3] S. Bespamyatnikh. Efficient algorithm for finding two larget empty circles. In *Proc. European Workshop on Computational Geometry*, page to appear, 1999.

[4] T.C. Biedl, E.D. Demaine, M.L. Demaine, A. Lubiw, and G.T. Toussaint. Hiding disks in folded polygons. In *Proc. 10th Canad. Conf. Comput. Geom.*, 1998.

[5] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[6] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes Comput. Sci.* Springer-Verlag, June 1991.

[7] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.

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

[9] C.-S. Shin, J.-H. Kim, S. K. Kim, and K.-Y. Chwa. Two-center problems for a convex polygon. In *Proc. European Symp. on Algorithm*, volume 1461 of *Lecture Notes Comput. Sci.*, pages 199–210. Springer-Verlag, 1998.

[10] T.-C. Yang, S.Y. Shin, and K.-Y. Chwa. Rolling discs and their applications. *J. of Design and Manufacturing*, 2:71–82, 1992.