Efficient algorithms for two-center problems for a convex polygon*

Sung Kwon Kim†
Dept. of Computer Engineering
Chung-Ang University
221 Huksuk-dong Dongjak-ku
Seoul 156–756
Republic of Korea
skkim@cau.ac.kr

Chan-Su Shin
Dept. of Computer Science
HKUST
Clear Water Bay
Hong Kong
cssin@cs.ust.hk

December 20, 1999

Abstract

We investigate some variants of the two-center problem in the plane. Let P be a convex polygon with n vertices. We wan to find two congruent closed disks whose union covers P (its boundary and interior) and whose radius is minimized. We also consider its discrete version with centers restricted to be at some vertices of P. Standard and discrete two-center problems are respectively solved in $O(n \log^3 n \ \log\log n)$ and $O(n \log^2 n)$ time. Furthermore, we can solve both of the standard and discrete two-center problems for a set of points that are in convex positions in $O(n \log^2 n)$ time.

Keywords: Geometric optimization, two centers, convex polygons

1 Introduction

Let A be a set of n points in the plane. The *standard two-center problem* for A is to cover A by a union of two congruent closed disks whose radius is as small as possible. The standard two-center problem has been studied extensively. Sharir [13] firstly presented a near-linear algorithm running in $O(n \log^9 n)$ time, and Eppstein [8] subsequently proposed a randomized algorithm with expected $O(n \log^2 n)$ running time. Chan [3] recently gave a randomized algorithm that runs in $O(n \log^2 n)$ time with high probability, as well as a deterministic algorithm that runs in $O(n \log^2 n (\log\log n)^2)$ time. As a variant, we consider the *discrete two-center problem* for A that finds two congruent closed disks whose union covers A and whose centers are at points of A. Recently, this problem was solved in $O(n^{4/3} \log^5 n)$ time by Agarwal et al. [1].

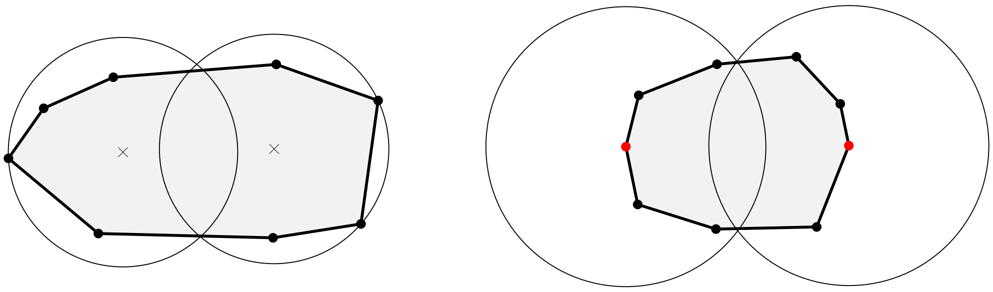
*This is an improved version of the paper *Two-Center Problems for a Convex Polygon*, presented at European Symposium on Algorithms (ESA), 1998.

† *Correspondence author*

	Standard 2-center		Discrete 2-center	
Convex polygon	$O(n \log^3 n \log\log n)$	Thm. 3	$O(n \log^2 n)$	Thm. 5
Points in convex positions	$O(n \log^2 n)$	Thm. 7	$O(n \log^2 n)$	Thm. 6
Points in arbitrary positions	$O(n \log^2 n\ (\log\log n)^2)$	[3]	$O(n^{4/3} \log^5 n)$	[1]

Table 1: The summary of the results.

In this paper, we consider some variants of the two-center problems. Let P be a convex polygon with n vertices in the plane. We wan to find two congruent closed disks whose union covers P (its boundary and interior) and whose radius is minimized. We also consider its *discrete* version with centers restricted to be at some vertices of P. See Figure 1. Compared with the two-center problems for points, differences are that (1) points to be covered by two disks are the vertices of P in convex positions (not in arbitrary positions) and (2) two disks should cover the edges of P as well as its vertices. By points in *convex positions*, we mean that the points form the vertices of a convex polygon. (1) suggests our problems are most likely easier than the standard point-set two-center problems, but (2) tells us that they could be more difficult.



(a) (b)

Figure 1: Two center problems for a convex polygon. (a) Standard problem. (b) Discrete problem.

We assume in this paper that the vertices of P are in *general circular position*, meaning that no four or more vertices are co-circular. Our results are summarized as follows; also see Table 1.

- We consider the **standard** two-center problem for a convex polygon of n vertices, and present an $O(n \log^3 n \log\log n)$ time algorithm (Theorem 3 in Section 2). The currently best known deterministic two-center algorithm for points runs in $O(n \log^2 n (\log\log n)^2)$ time [3].
- We next consider the **discrete** two-center problem for a convex polygon of n vertices, and show that the problem can be solved in $O(n \log^2 n)$ time (Theorem 5 in Section 3). This algorithm can easily be modified to solve the discrete two-center problem for a set of n points in convex positions in $O(n \log^2 n)$ time (Theorem 6 in Section 3). The discrete two-center problem for a set of points in arbitrary positions is solved in $O(n^{4/3} \log^5 n)$ time [1].
- We also consider the standard two-center problem for a set of n points in convex positions and solve in $O(n \log^2 n)$ time (Theorem 7 in Section 4).

Most of our algorithms presented in this paper are based on parametric search technique proposed by Megiddo [12]. Parametric search is an optimization technique which can be applied in situations where we seek a minimum parameter r^* satisfying the *monotone* condition that is met by all $r \geq r^*$ but not by any $r < r^*$. The strategy of parametric search is to design efficient sequential and parallel algorithms, A_s and A_p, for the corresponding decision problem: decide whether a given parameter $r \geq r^*$, and executes A_p "generically" without specifying the value of r, with the intention of simulating its execution at the unknown value r^*. If A_s takes $O(T_s)$ time, and A_p takes $O(T_p)$ time using $O(P)$ processors, parametric search results in a sequential algorithm for the optimization problem with running time $O(PT_p + T_sT_p \log P)$. Cole [7] described an improvement technique that achieves $O(PT_p + T_s(T_p + \log P))$ time, assuming that the parallel algorithm satisfies a "bounded fan-in/out" requirement.

2 Algorithm for standard two-center problem for a convex polygon

Let P be a convex polygon with n vertices. The vertices are numbered $0, 1, \ldots, n-1$[1] counterclockwise and an edge is denoted by specifying its two end points, e.g., $(i, i+1)$. Let $\langle a, b\rangle$ be the counterclockwise sequence of vertices $a, a+1, \ldots, b$, if $a \leq b$, and $a, a+1, \ldots, n-1, 0, \ldots, b$, otherwise.

Let r^* be the minimum value such that two disks of radius r^* cover P. Since parametric search technique will be employed, we will give a sequential algorithm and a parallel algorithm that decides, given a value $r > 0$, whether $r \geq r^*$, i.e., whether two disks of radius r can be drawn so that their union covers P.

2.1 Sequential decision algorithm

Our decision algorithm starts with drawing for each i a disk D_i of radius r with center at vertex i. Define

$$\mathcal{I}(\langle a, b\rangle) = \bigcap_{i=a}^{b} D_i.$$

Define m_i to be the index such that $\mathcal{I}(\langle i, m_i\rangle) \neq \emptyset$ and $\mathcal{I}(\langle i, m_i + 1\rangle) = \emptyset$. The index m_i is the "counterclockwise farthest" vertex from i so that a disk of radius r can include the vertices in $\langle i, m_i\rangle$. Note that the m_i's are monotone increasing, i.e., there is no pair of $\langle i, m_i\rangle$ and $\langle j, m_j\rangle$ such that $\langle i, m_i\rangle \subset \langle j, m_j\rangle$.

Let $\mathcal{J}_i = \mathcal{I}(\langle i, m_i\rangle)$. Then any point in $\mathcal{J}_i$ can be the center of a disk of radius r that covers $\langle i, m_i\rangle$. We check each of the following cases.

(i) If there are two vertices i and j such that $i, m_i \in \langle j, m_j\rangle$ and $j, m_j \in \langle i, m_i\rangle$, then any two disks of radius r, one with center in $\mathcal{J}_i$ and the other with center in $\mathcal{J}_j$, can cover P, and the decision algorithm returns "yes". To check for a fixed i whether there is an index j such that $i, m_i \in \langle j, m_j\rangle$ and $j, m_j \in \langle i, m_i\rangle$, it is sufficient to consider the case of $j = m_i$ only, due to the monotonicity of m-values.

(ii) If there are i and j such that $m_j + 1 = i$ and $j \in \langle i, m_i\rangle$, then $\langle i, m_j\langle$ is guaranteed to be covered by two disks of radius r with centers in $\mathcal{J}_i$ and in $\mathcal{J}_j$ and the edge (m_j, i) of P is "missing". So, we need check whether it can be covered by two disks of radius r with centers in $\mathcal{J}_i$ and in $\mathcal{J}_j$.

[1] All vertex indices are taken modulo n.

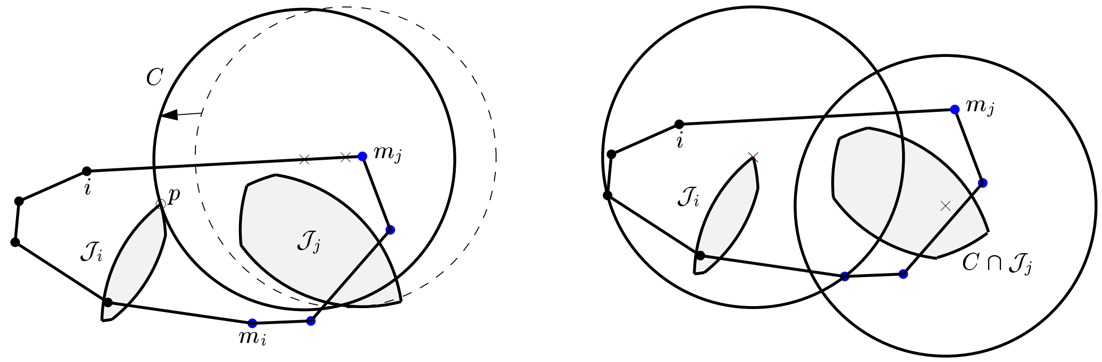


Figure 2: An illustration of case (ii).

To do this, as illustrated in Figure 2(a), find a disk C of radius r with center on the edge (m_j, i) and touching $\mathcal{J}_i$ at a point p and checks whether C intersects $\mathcal{J}_j$. One disk of radius r with center p and the other with center at any point in $C \cap \mathcal{J}_j$ together cover the edge (m_j, i) as well as the other boundary of P (see Figure 2(b)). If such point p can be found and $C \cap \mathcal{J}_j \neq \emptyset$, then the decision algorithm returns "yes". Again, for a fixed i, it does not need to consider all j's such that $m_j = i - 1$ and $j \in \langle i, m_i \rangle$. It is easy to see that considering the case of $j = m_i$ is sufficient.

(iii) Finally, if there are i and j such that $m_i + 1 = j$ and $m_j + 1 = i$, then two edges (m_i, j) and (m_j, i) are "missing". To decide whether these two edges can simultaneously be covered by two disks of radius r with centers in $\mathcal{J}_i$ and in $\mathcal{J}_j$, find as above a disk C_1 of radius r with center on (m_j, i) and touching $\mathcal{J}_j$, and a disk C_2 of radius r with center on (m_j, i) and touching $\mathcal{J}_i$. Similarly, find two disk C_1', C_2' of radius r with centers on (m_i, j) touching $\mathcal{J}_j$ and $\mathcal{J}_i$, respectively. Now, check whether both $C_1 \cap C_1' \cap \mathcal{J}_j$ and $C_2 \cap C_2' \cap \mathcal{J}_i$ are nonempty. If both are nonempty, two disks of radius r, one with center in $C_1 \cap C_1' \cap \mathcal{J}_j$ and the other with center in $C_2 \cap C_2' \cap \mathcal{J}_i$, can cover both edges as well as the other boundary of P, and thus the decision algorithm returns "yes".

The decision returns "no" if none of the three cases above returns "yes".

Implementation. Compute m_0 and $\mathcal{J}_0$, and then $\mathcal{J}_{m_0}$ and $\mathcal{J}_{m_0+1}$. For $i > 0$, $\mathcal{J}_i$, $\mathcal{J}_{m_i}$, and $\mathcal{J}_{m_i+1}$ can be obtained from $\mathcal{J}_{i-1}$, $\mathcal{J}_{m_{i-1}}$ and $\mathcal{J}_{m_{i-1}+1}$, by adapting the algorithm in [9] for maintaining convex hulls of a simple chain while points are dynamically inserted into and deleted from the ends of the chain. An intermixed sequence of $O(\ell)$ insertions and deletions on chains with a total of $O(\ell)$ vertices can be processed in time $O(\ell \log \ell)$ [9].

Whenever $\mathcal{J}_i$, $\mathcal{J}_{m_i}$, and $\mathcal{J}_{m_i+1}$ are obtained, we check whether any of three cases (i), (ii) and (iii) returns "yes". Each of these checks can be done in $O(\log n)$ time, provided that $\mathcal{J}_i$, $\mathcal{J}_{m_i}$, and $\mathcal{J}_{m_i+1}$ are given. For this we need a constant number of the following operations.

(O1) Find a touching point between the intersection of $O(n)$ congruent disks and another congruent disk with center on a specific edge of P.

(O2) Decide whether a congruent disk intersects an intersection of $O(n)$ congruent disks. If yes, find the intersection points on the boundary.

Both (O1) and (O2) can be done in $O(\log n)$ time by simply performing binary searches on the boundary of the intersection of $O(n)$ congruent disks. We assume here that the boundaries of the intersections are stored in arrays or balanced trees for binary searches to be applicable.

As a whole, we have the following theorem.

Theorem 1 *Given a convex polygon P with n vertices and a value $r > 0$, we can decide in $O(n \log n)$ time whether two disks of radius r can cover P.*

2.2 Parallel decision algorithm

Our parallel algorithm builds a segment tree on the sequence, $D_0, \ldots, D_{n-1}$, of radius r disks centered at vertices. Each internal node[2] of the tree corresponds to a *canonical subsequence* of the vertices that are assigned to the leaves of the subtree rooted at the node. For each canonical subsequence S, compute the intersection of the disks, $\mathcal{I}(S) = \cap_{i \in S} D_i$. The boundary of $\mathcal{I}(S)$ is divided into two (upper and lower) chains by its leftmost and rightmost points, and these two chains are stored into two arrays of arcs[3] sorted according to their x-coordinates.

Constructing the segment tree, computing $\mathcal{I}(S)$ for each canonical subsequence S, and storing their chains into arrays can be carried out in $O(\log n)$ time using $O(n \log n)$ processors, as explained in [3].

Computing m_i for each i can be done in divide-and-conquer manner. Compute $m_{\lfloor n/2 \rfloor}$ by noting $m_{\lfloor n/2 \rfloor} = \ell$ such that $\mathcal{I}(\langle \lfloor n/2 \rfloor, \ell \rangle) \neq \emptyset$ and $\mathcal{I}(\langle \lfloor n/2 \rfloor, \ell + 1 \rangle) = \emptyset$. Then due to the monotonicity of the m-values, $m_i \in \langle 0, m_{\lfloor n/2 \rfloor} \rangle$ for $i \in \langle 0, \lfloor n/2 \rfloor - 1 \rangle$, and $m_i \in \langle m_{\lfloor n/2 \rfloor}, n-1 \rangle$ for $i \in \langle \lfloor n/2 \rfloor + 1, n-1 \rangle$. Thus, they can be recursively obtained. Our parallel algorithm for this step performs $O(\log n)$ recursions each consisting of $O(n)$ decisions on the emptiness of $\mathcal{I}(\langle a, b \rangle)$ for different a and b. Since, given a and b, whether $\mathcal{I}(\langle a, b \rangle) = \emptyset$ can be decided in $O(\log n \log \log n)$ time using $O(\log n)$ processors [3], the whole of this step takes $O(\log^2 n \log \log n)$ time using $O(n \log n)$ processors.

Checking the first case (i) in parallel is easy. For each i, decide whether $m_{m_i} \in \langle i, m_i \rangle$. The algorithm returns "yes" if any one of these n decisions is true.

For case (ii), collect all i's such that $m_{m_i} = i - 1$, and for each such i decide whether edge $(i-1, i)$ can be covered by two disks of radius r with centers in $\mathcal{J}_i$ and in $\mathcal{J}_{m_i}$.

For case (iii), collect all i's such that $m_{m_i+1} = i - 1$, and for each such i decide whether both edges $(m_i, m_i + 1)$ and $(i-1, i)$ can be simultaneously covered by two disks of radius r with centers in $\mathcal{J}_i$ and in $\mathcal{J}_{m_i+1}$.

For any of cases (ii) and (iii), as in the sequential algorithm, our parallel algorithm also needs to compute $\mathcal{J}_i$ for each i. Computing all $\mathcal{J}_i$'s and storing them explicitly requires $\Omega(n^2)$ work and space, which is far from the bounds of our parallel algorithm. Instead, whenever a $\mathcal{J}_i$ is needed, we compute it from the canonical subsequences of $\langle i, m_i \rangle$ with the help of the segment tree.

Assume that $\langle i, m_i \rangle$ is a union of $k = O(\log n)$ canonical subsequences $S_1, \ldots, S_k$, and also assume that $S_1, \ldots, S_k$ are indexed counterclockwise so that $i \in S_1, m_i \in S_k$, and S_ℓ and $S_{\ell+1}$ for $1 \leq \ell \leq k-1$ appear consecutively on the boundary of P. Then, $\mathcal{J}_i = \mathcal{I}(S_1) \cap \ldots \cap \mathcal{I}(S_k)$. If $i > m_i$, then $\langle i, m_i \rangle$ is divided into $\langle i, n-1 \rangle$ and $\langle 0, m_i \rangle$. $\langle i, m_i \rangle$ is a union of the canonical subsequences for $\langle i, n-1 \rangle$ and the canonical subsequences for $\langle 0, m_i \rangle$.

[2] To avoid confusion, *nodes* instead of vertices will be used in trees.

[3] Instead of edges, we use *arcs* to denote sides of an intersection of disks.

Let $\beta(\mathcal{J}_i)$ be the upper chain of $\mathcal{J}_i$ and U_j be the upper chain of $\mathcal{I}(S_j)$ for $1 \leq j \leq k$. We show how $\beta(\mathcal{J}_i)$ is constructed from the U_j. The computation of its lower chain is completely symmetric.

Build a minimum-height binary tree whose leaves correspond to $U_1, \ldots, U_k$ in a left-to-right order. Let v be a node of the tree and its canonical subsequence be $\langle a, b \rangle$. Let $\mathcal{I}_v = \mathcal{I}(S_a) \cap \ldots \cap \mathcal{I}(S_b)$ and $\beta(\mathcal{I}_v)$ be its upper chain. Then, $\beta(\mathcal{I}_v)$ consists of some subchains of $U_a, U_{a+1}, \ldots, U_b$[4] and thus can be represented by a set of tuples, where each tuple says which subchain of which U_j contributes to $\beta(\mathcal{I}_v)$.

In a bottom-up fashion, compute $\beta(\mathcal{I}_v)$ for each node v of the minimum-height binary tree. For each leaf v corresponding to U_j, $\beta(\mathcal{I}_v) = \{(j, 1, x^-, x^+, 1, |U_j|)\}$, where x^- (resp., x^+) is the x-coordinate of the leftmost (resp., rightmost) point of U_j. For each internal node v, $\beta(\mathcal{I}_v) = \{(j_1, p_1, x_1^-, x_1^+, r_1^-, r_1^+), \ldots, (j_t, p_t, x_t^-, x_t^+, r_t^-, r_t^+)\}$, where t is the number of subchains of U_j's that consist of $\beta(\mathcal{I}_v)$. Each tuple represents a subchain – which U_j it originally comes from, the rank of its leftmost arc in its original U_j, its x-range, and the ranks of its leftmost and rightmost arcs in $\beta(\mathcal{I}_v)$. For example, the first tuple $\beta(\mathcal{I}_v)$ means the following three things:

1. the subchain of U_{j_1} in the x-range $[x_1^-, x_1^+]$ appears in $\beta(\mathcal{I}_v)$;
2. its leftmost (resp., rightmost) arc is bounded by a vertical line $x = x_1^-$ (resp., $x = x_1^+$) and its leftmost arc is a part of the p_1-th arc in U_{j_1}; and
3. its leftmost (resp., rightmost) arc becomes the r_1^--th (resp., r_1^+-th) arc in $\beta(\mathcal{I}_v)$.

Lemma 1 *Assume that v has two children w and z. Given $\beta(\mathcal{I}_w)$ and $\beta(\mathcal{I}_z)$, $\beta(\mathcal{I}_v)$ can be obtained in $O(\log n \cdot (|\beta(\mathcal{I}_w)| + |\beta(\mathcal{I}_z)| + \log n))$ sequential time.*

Proof: Let $f = \beta(\mathcal{I}_w)$ and $g = \beta(\mathcal{I}_z)$. Let S^w and S^z be the canonical subsequences of w and z, respectively. In other words, $\mathcal{I}_w = \cap_{i \in S^w} D_i$ and $\mathcal{I}_z = \cap_{i \in S^z} D_i$. Let $|S^w|$ and $|S^z|$ be the number of vertices in S^w and S^z. Let $[x^-, x^+]$ be the common x-range of f and g. The intersection points, if any, of f and g must be in the vertical strip bounded by two lines $x = x^-$ and $x = x^+$. So, we may assume that f and g are functions over $[x^-, x^+]$. Note that f and g consist of several arcs of radius r. Lemma 1 will be proved through a series of lemmas.

Lemma 2 *f and g intersect at most twice.*

Proof: If they intersect more than twice, then we can find four arcs $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, in right-to-left order, in $\beta(\mathcal{I}_v)$ such that either $\alpha_1, \alpha_3 \in f$ and $\alpha_2, \alpha_4 \in g$, or $\alpha_1, \alpha_3 \in g$ and $\alpha_2, \alpha_4 \in f$, that is, the arcs of f and g appear alternately. This is impossible as the centers of the discs we are dealing with are at vertices of a convex polygon, and w and z are the children of the same parent v. (S^w and S^z appear consecutively on the boundary of P, and $S^w \cup S^z$ is the canonical subsequence of v.) **QED**

Since both f and g are upper chains of the intersection of congruent disks, they are upward convex with one highest point. Define $h(x) = f(x) - g(x)$ over $[x^-, x^+]$.

Lemma 3 *If both f and g are either increasing or decreasing over $[x^-, x^+]$, $f(x^-) > g(x^-)$, and $|S^w| \geq 3$ and $|S^z| \geq 3$, then h over $[x^-, x^+]$ is a downward convex function with one lowest point.*

[4] A subchain of U_j is a part of U_j between any two points on it.

Proof: Assume that both f and g are increasing and $f(x^-) > g(x^-)$. Proof for the case of both decreasing is symmetric. We start with noting that for any canonical subsequence S, due to the convexity of $\beta(\mathcal{I}(S))$, when we follow its arcs clockwise, (resp., counterclockwise,) their corresponding centers appear on the boundary of P clockwise. (resp., counterclockwise.)

Since both f and g are increasing, each arc of f and g is a subarc of a second quadrant.[5] ¿From geometric properties of congruent circles, the following facts are true. In the following an arc of f and g is denoted by α_i by specifying its center vertex i. For $\alpha_i \in f$ and $\alpha_j \in g$, their common x-range is denoted by $[x_{ij}^-, x_{ij}^+]$.

(i) For any two arcs α_i, α_ℓ of f such that α_ℓ precedes α_i counterclockwise, i lies in the first quadrant of ℓ. The same is true for any pair of arcs of g.

(ii) If $\alpha_i \in f$ and $\alpha_j \in g$ do not intersect and α_i is strictly above α_j over $[x_{ij}^-, x_{ij}^+]$, then h over $[x_{ij}^-, x_{ij}^+]$ is either increasing or decreasing. If it is increasing, then i lies in the first quadrant of j, and, otherwise, i lies to the left of j.

(iii) If $\alpha_i \in f$ and $\alpha_j \in g$ intersect at a point and $f(x_{ij}^-) > g(x_{ij}^-)$, then j lies in the first quadrant of i.

We now have three cases depending on the number of intersections between f and g. Note that there are at most two intersections by Lemma 2.

Case 1: f and g do not intersect. In this case, h will be proven to be either increasing or decreasing. For a contradiction, suppose that there is an x-coordinate x^* such that h is increasing over $[x^-, x^*]$ and then starts decreasing at x^*. Let m be the vertical line $x = x^*$. Suppose that m intersects f at point p and g at point q. One of p and q should be a vertex of f and g. In other words, m does not properly intersect both arcs of f and g. If m do properly intersect α_i of f and α_j of g, then h over $[x_{ij}^-, x_{ij}^+]$ is either increasing or decreasing by fact (ii).

1. *q is a vertex of g.* Assume that p is a point of α_i of f and q is the right end vertex of α_j and the left end vertex of α_ℓ of g. By fact (i), j lies in the first quadrant of ℓ. By fact (ii), i lies in the first quadrant of j, and i lies to the left of ℓ. This is impossible.

2. *p is a vertex of f.* Assume that q is a point of α_j of g, and p is the right end vertex of α_i and the left end vertex of α_ℓ of f. So, $|S^w| \geq 2$. By fact (i), i lies in the first quadrant of ℓ. By fact (ii), i lies in the first quadrant of j, and ℓ lies to the left of j. In other words, j lies in the vertical strip bounded by two vertical lines passing through i and ℓ, and below the horizontal line passing through i (see Figure 3(a)). Note that $\langle \ell, i \rangle$ is a convex chain of P.

   - *j lies below $\langle \ell, i \rangle$.* Then, $S^w \cup S^z$ is no longer convex as $\langle \ell, i \rangle \subseteq S^w$ and $j \in S^z$ (see Figure 3(a)). Since $|S^w| \geq 3$ and it is a convex chain, we know that the region above it is the interior of P. In case $|S^w| = 2$, $\langle \ell, i \rangle$ consists of a single edge; thus either the region above or below it might be the interior of P depending on where the edge locates, and $S^w \cup S^z$ still could be convex.
   - *j lies above $\langle \ell, i \rangle$.* Let D be a disk of radius r with center at p. Then i and ℓ lie on the boundary of D and the other vertices of $\langle \ell, i \rangle$ lie all in the interior of D. Thus D

[5] A point p partitions the plane into four quadrants by the vertical and horizontal lines passing through it, namely, the ***first, second, third,*** and ***fourth*** ones, counterclockwise starting at the upper right one. Four quadrants of a circle is similarly defined by the vertical and horizontal lines passing through its center.

also contains j (see Figure 3(b)). A disk of radius r with center at j (α_j is a part of its boundary) will contain p, and thus q will be above p, which is a contradiction.

So, there is no such x^* and h is increasing over $[x^-, x^+]$. The case of h decreasing over $[x^-, x^+]$ can be handled in a symmetric way.

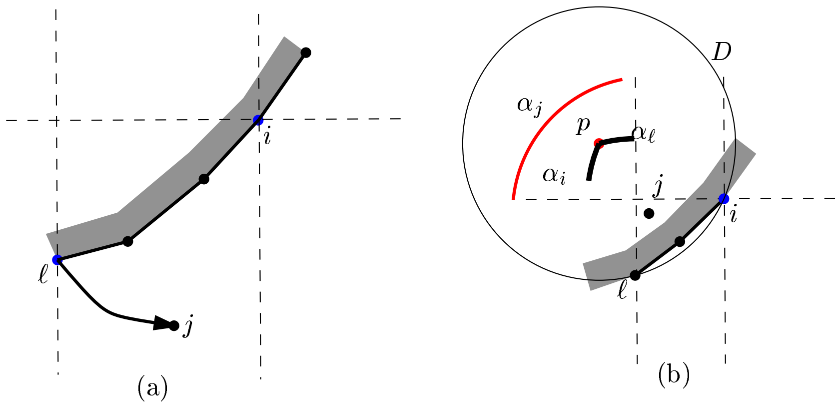


Figure 3: Case 1.

Case 2: f and g intersect at one point z. Let $z = \alpha_i \cap \alpha_j$ for $\alpha_i \in f$ and $\alpha_j \in g$, and let x^z be the x-coordinate of z. Using arguments similar to the one in Case 1, we can show that h is decreasing over $[x^-, x^z]$, and it is also decreasing over $[x^z, x^+]$.

Case 3: f and g intersect at two points z_1 and z_2. Let x^{z_1} and x^{z_2} be the x-coordinates of z_1 and z_2, respectively. Applying arguments similar to the one used in Case 1, we can easily show that h is decreasing over $[x^-, x^{z_1}]$, and it is increasing over $[x^{z_2}, x^+]$. To show that h over $[x^{z_1}, x^{z_2}]$ is decreasing and then increasing (i.e., is downward convex), let x^* be an x-coordinate such that h is decreasing over $[x^{z_1}, x^*]$ and then starts increasing at x^*. We can easily show by applying an argument similar to the one used in Case 1, that h over $[x^*, x^{z_2}]$ is increasing only. Thus h is a downward convex function with one lowest point whose x-coordinate is x^*. **QED**

Lemma 4 *If both f and g are either increasing or decreasing over $[x^-, x^+]$, then their intersections, if any, can be found in $O(\log n \cdot (|\beta(\mathcal{I}_w)| + |\beta(\mathcal{I}_z)| + \log n))$ sequential time.*

Proof: Assume that f and g are increasing and $f(x^-) > g(x^-)$. Proof for the case of both decreasing is symmetric. Let $|f|$ and $|g|$ be the number of arcs consisting of f and g, respectively. Assume that $|S^w| \geq 3$ and $|S^z| \geq 3$. The case when either $|S^w| \leq 2$ or $|S^z| \leq 2$ will be mentioned at the end of the proof. By Lemma 3, $h(x)$ is a downward convex function with one lowest point over $[x^-, x^+]$. The roots of $h(x) = 0$ correspond to the intersection points of f and g, and if $h(x) = 0$ has no root, then f and g do not intersect. Let x^* be the x-coordinate of the lowest point of $h(x)$. f and g intersect iff $h(x^*) \leq 0$. If $h(x^*) = 0$ then they touch each other. If $h(x^*) < 0$, then divide $[x^-, x^+]$ into two subranges $[x^-, x^*]$ and $[x^*, x^+]$, and find a root of $h(x) = 0$, if any, in each subrange.

To find x^*, we do binary search on the arcs of f with the following idea: Find the medium-indexed arc α of f, and decide whether the x-range of α contains x^*, or h over the x-range is increasing or decreasing. Depending on the decision, our algorithm either finds x^*, or removes a half of the arcs of f from further consideration and repeats with the remaining half.

Let $q = \lfloor |f|/2 \rfloor$. Find a tuple in $\beta(\mathcal{I}_w)$ with $r_i^- \leq q \leq r_i^+$ by checking the tuples one by one. Then, the q-th arc in f is a subarc of the $(q - r_i^- + p_i)$-th arc in U_{j_i}.

Let α be the q-th arc in f and $[a^-, a^+]$ be its x-range. Of course, $[a^-, a^+] \subseteq [x^-, x^+]$. Let a^* be the x-coordinate of the lowest point of $h(x)$ over $[a^-, a^+]$. If $a^* = a^-$, then $h(x)$ over $[a^-, a^+]$ is increasing and thus the second half of f can be discarded. If $a^* = a^+$, then $h(x)$ over $[a^-, a^+]$ is decreasing, and thus the first half of f can be discarded. Otherwise, $x^* = a^*$.

Given $\alpha \in f$ and its x-range $[a^-, a^+]$, to find a^* we do binary search on the arcs of g whose x-ranges are in $[a^-, a^+]$. Wlog, let $|g|$ be the number of arcs in g whose x-ranges are in $[a^-, a^+]$. Let $q' = \lfloor |g|/2 \rfloor$. Then we can find from $\beta(\mathcal{I}_z)$ a tuple containing information about the q'-th arc of g. Let α' be the q'-th arc of g, and $[a'^-, a'^+]$ be its x-range. Of course, $[a'^-, a'^+] \subseteq [a^-, a^+]$. Find over $[a'^-, a'^+]$ the lowest point of $h(x)$ and compute its x-coordinate a'^*. Since we have two arcs α and α', it is easy to find a'^* using two circle equations of radius r.

If $a'^* = a'^-$, then $h(x)$ over $[a'^-, a'^+]$ is increasing, and the second half of g can be discarded. If $a'^* = a'^+$, then $h(x)$ over $[a^-, a^+]$ is decreasing, and the first half of g can be discarded. Otherwise, $a^* = a'^*$.

We analyze the time complexity of our "double" binary search. For a fixed q, to find the q-th arc we scan $\beta(\mathcal{I}_w)$, taking $O(|\beta(\mathcal{I}_w)|)$ time, and to compute a^* we do binary search over g, taking $O(|\beta(\mathcal{I}_z)| + \log n)$ time. Since $O(\log n)$ different q's are searched, x^* can be found in $O(\log n \cdot (|\beta(\mathcal{I}_w)| + |\beta(\mathcal{I}_z)| + \log n))$ time.

Finding an x with $h(x) = 0$ over $[x^-, x^*]$ and over $[x^*, x^+]$ can be done by applying "double" binary search similar to the one used above.

Finally, if $|S^w| \leq 2$, then the intersections between f and g can directly be computed by using a constant number of operations (O2) or their modifications. Since, here, $\beta(\mathcal{I}_z)$ is the upper boundary of $O(n)$ congruent disks and is represented as a set of tuples, computing the intersections between f and g takes $O(|\beta(\mathcal{I}_z)| \cdot \log n)$ time. The case of $|S^z| \leq 2$ can be handled symmetrically. **QED**

Lemma 5 *If one of f and g is increasing and the other is decreasing over $[x^-, x^+]$, then their intersection, if any, can be found in $O(\log n \cdot (|\beta(\mathcal{I}_w)| + |\beta(\mathcal{I}_z)| + \log n))$ sequential time.*

Proof: If either $f(x^-) > g(x^-)$ and $f(x^+) > g(x^+)$ or $f(x^-) < g(x^-)$ and $f(x^+) < g(x^+)$, then they do not intersect. Otherwise, they intersect exactly once, and in this case h is either increasing or decreasing. Its root of $h(x) = 0$ can be found by doing "double" binary search similar to the one in Lemma 4. **QED**

To compute the intersection points between f and g, find x^f and x^g, the x-coordinates of the highest points of f and g, respectively.

Lemma 6 *x^f can be computed in $O(|\beta(\mathcal{I}_w)| \cdot \log n)$, and x^g can be computed in $O(|\beta(\mathcal{I}_z)| \cdot \log n)$.*

Proof: This is a simple modification of the "double" binary search in the proof of Lemma 4. **QED**

Wlog, assume that $x^f < x^g$. Then, $[x^-, x^+]$ is divided into three subranges $[x^-, x^f]$, $[x^f, x^g]$, and $[x^g, x^+]$. In each subrange, f is either increasing or decreasing and the same is true for g. If both are either increasing or decreasing, then Lemma 4 applies, and if one is increasing and the other is decreasing, then Lemma 5 applies.

After locating the intersection points between f and g, it is easy to determine which tuples from $\beta(\mathcal{I}_w)$ and $\beta(\mathcal{I}_z)$ are eligible to be in $\beta(\mathcal{I}_v)$. Thus, the tuples of $\beta(\mathcal{I}_v)$ can be computed in additional $O(|\beta(\mathcal{I}_w)| + |\beta(\mathcal{I}_z)|)$ time.

This completes the proof of Lemma 1. **QED**

Given all $\beta(\mathcal{I}_w)$ for the nodes w of some level of the minimum-height tree, if k processors are available, then all $\beta(\mathcal{I}_v)$ for the nodes v of the next level (toward the root) can be obtained in $O(\log n \cdot (|\beta(\mathcal{I}_w)| + \log n)) = O((k + \log n)\log n)$ time by Lemma 1. Note that $|\beta(\mathcal{I}_w)| = O(k)$ for any node w. $|\beta(\mathcal{I}_w)|$ is the number of tuples, not the number of arcs in it. The minimum-height tree with k leaves has $O(\log k)$ levels.

Lemma 7 *For any i, $\beta(\mathcal{J}_i)$, in the form of a set of tuples, can be constructed in $O((k+\log n)\log k \log n)$ time using $O(k)$ processors.*

Back to checking case (ii) or case (iii), assign $k = O(\log n)$ processors to each i satisfying the condition of (ii) or (iii), and construct $\mathcal{J}_i$, in the form of two sets of tuples, one for its upper chain and the other for its lower chain, in $O(\log^2 n \log\log n)$ time by Lemma 7. Since there are at most n such i's, we need a total of $O(n \log n)$ processors.

The remaining part of (ii) or (iii) can be done using a constant number of operations (O1) and (O2) on $\mathcal{J}_i$. Each of (O1) and (O2), using the two sets of tuples, can now be executed in $O(\log^2 n)$ sequential time.

As a result, we have the theorem.

Theorem 2 *Given a convex polygon P with n vertices and a value $r > 0$, we can decide in $O(\log^2 n \log\log n)$ time using $O(n\log n)$ processors whether two disks of radius r cover P.*

Combining Theorems 1 and 2, we have proved the theorem.

Theorem 3 *Given a convex polygon P with n vertices, we can compute the minimum radius r^* in $O(n\log^3 n \log\log n)$ time such that two disks of radius r^* cover P.*

Proof: By parametric search technique, we immediately have an $O(n\log^4 n \log\log n)$ time algorithm. Our parallel algorithm can easily be made to satisfy the Cole's requirement [7]. The time complexity is now reduced to $O(n\log^3 n \log\log n)$. **QED**

3 Algorithm for discrete two-center problem for a convex polygon

Let P be a convex polygon with n vertices. We want to find the minimum value r^* and two vertices i, j such that two disks of radius r^* with centers i and j cover P. Since again parametric search technique will be employed, we will give a sequential algorithm and a parallel algorithm that decides, given a value $r > 0$, whether $r \geq r^*$.

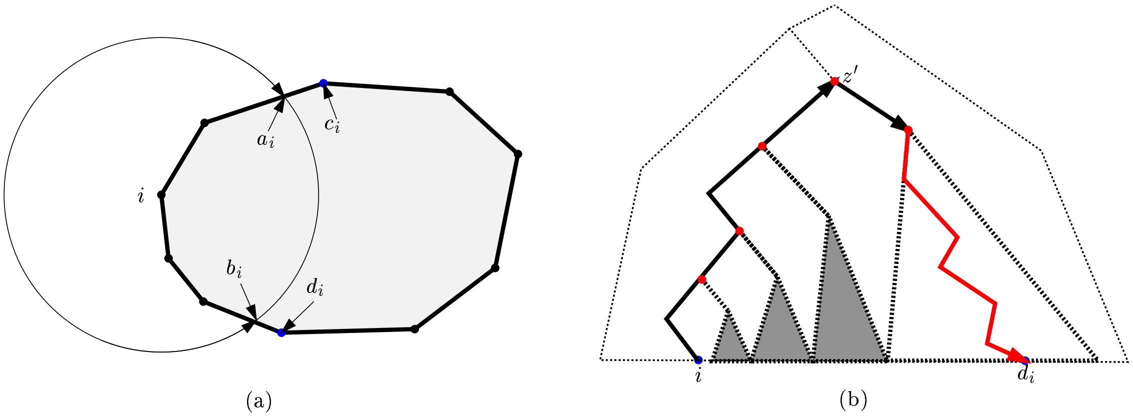


Figure 4: (a) Definitions of a_i, b_i, c_i, and d_i. (b) Searching for d_i.

3.1 Sequential decision algorithm

Let ∂P denote the boundary of P. For two points a, b on ∂P, $\langle\langle a, b\rangle\rangle$ denotes the subchain of ∂P that walks counterclockwise starting at a and ending at b.

Given $r > 0$, draw a disk D_i of radius r with center at vertex i for each i. Then, $D_i \cap \partial P$ may be disconnected and thus may consist of several subchains of P. One of them contains i, which will be denoted by $\partial_i = \langle\langle a_i, b_i\rangle\rangle$ (see Figure 4(a)).

Lemma 8 *$r \geq r^*$ iff there exist two vertices i and j such that $\partial P = \partial_i \cup \partial_j$.*

Proof: If there exist two such vertices, then clearly $P \subseteq D_i \cup D_j$ and thus $r \geq r^*$.

Now suppose that P is optimally covered by two disks D, D' of radius r^*. $D \cap \partial P$ consists of several subchains of P, and let $\langle\langle a, b\rangle\rangle$ be the one containing the center of D. Similarly, $D' \cap \partial P$ consists of several subchains of P, and let $\langle\langle a', b'\rangle\rangle$ be the one containing the center of D'.

We will show that $\langle\langle a, b\rangle\rangle \cup \langle\langle a', b'\rangle\rangle = \partial P$. Suppose that a point on ∂P is not in $\langle\langle a, b\rangle\rangle \cup \langle\langle a', b'\rangle\rangle$. Wlog, assume that the point is in $\langle\langle b', a\rangle\rangle$. Then there exist a point z in $\langle\langle b', a\rangle\rangle$ near to b' that is not in D'. Similarly, there exists another point z' in $\langle\langle b', a\rangle\rangle$ near to a that is not in D. So, $z \in D$ and $z' \in D'$ as $P \subseteq D \cup D'$. Then the boundaries of D and D' intersect at least four times, which is a contradiction. Thus, $\langle\langle a, b\rangle\rangle \cup \langle\langle a', b'\rangle\rangle = \partial P$. **QED**

Computing the end points a_i, b_i of ∂_i for each i can be done by simulating sequentially our parallel algorithm in Section 3.2. The simulation takes $O(n \log n)$ time for preprocessing and for a query i answers a_i and b_i in $O(\log n)$ time. Alternatively, the algorithm in [6] can be used. It, given $r > 0$, preprocesses P in $O(n \log n)$ time, and, for a query point c, one can find in $O(\log n)$ the first (clockwise and counterclockwise) hit point on ∂P when a circle of radius r with center at c is drawn.

After having ∂_i for all i, by Lemma 8 we need to decide whether there exist two vertices i and j such that $\partial P = \partial_i \cup \partial_j$. This is an application of *minimum circle cover* algorithm in [2, 10, 11] that runs in $O(n \log n)$ time. In the minimum circle cover problem, we are given a set arcs of a circle and want to find the minimum number of arcs that covers the circles (i.e., whose union is the circle).

Theorem 4 *Given a convex polygon P with n vertices and a value $r > 0$, we can decide in $O(n \log n)$ time whether two disks of radius r with centers at vertices of P can cover P.*

3.2 Parallel decision algorithm

Our parallel algorithm first computes ∂_i for each i by locating a_i and b_i. Build a segment tree on the sequence of radius r disks centered at vertices, $D_0, \ldots, D_{n-1}, D_0, \ldots, D_{n-2}$. For each node v with canonical subsequence S, compute the intersection of the disks, $\mathcal{I}_v = \cap_{i \in S} D_i$. The boundary of $\mathcal{I}_v$ is divided into two (upper and lower) chains by its leftmost and rightmost points, and the upper and lower chains are stored into two arrays, $\beta^+(\mathcal{I}_v)$ and $\beta^-(\mathcal{I}_v)$, respectively, of arcs sorted according to their x-coordinates. Given a point q, whether $q \in \mathcal{I}_v$ can be decided by doing binary searches on $\beta^+(I_v)$ and on $\beta^-(I_v)$.

To compute $\langle\langle a_i, b_i \rangle\rangle$, it is sufficient to find the minimal $\langle\langle c_i, d_i \rangle\rangle \supseteq \langle\langle a_i, b_i \rangle\rangle$, where c_i and d_i are vertices of P. See Figure 4(a).

For a given vertex i, d_i can be searched using the segment tree as follows (see Figure 4(b)): Choose the left one from the two D_i's at the leaves of the tree. We go upward along the leaf-to-root path starting at the chosen D_i, and locate the first node on the path with a right child z' satisfying $i \in \mathcal{I}_{z'}$. Starting at z', we now search downward. At a node with left child w and right child z, search under the following rule: Go to z if $i \in \mathcal{I}_w$, and go to w, otherwise. It is easy to verify that when search is over at a leaf v, we have $d_i = v$.

Searching c_i is completely symmetric.

Implementation. Constructing the segment tree and computing $\beta^+(\mathcal{I}_v)$ and $\beta^-(\mathcal{I}_v)$ for each v can be carried out in $O(\log n)$ time using $O(n \log n)$ processors [3] as in Section 2.2.

For a fixed i, c_i and d_i can be found in $O(\log^2 n)$ sequential time as we visit $O(\log n)$ nodes and at each node v we spend $O(\log n)$ time doing binary search to decide whether $i \in \mathcal{I}_v$. This type of search (i.e., searching the same value in several sorted lists along a path of a binary tree) can be improved by an application of fractional cascading [4, 5], saving a logarithmic time to achieve $O(\log n)$ search time.

From c_i and d_i, we compute $\partial_i = \langle\langle a_i, b_i \rangle\rangle$ by computing the intersection points between a circle of radius r with center i and two edges $(c_i, c_i + 1), (d_i - 1, d_i)$. With ∂_i for all i, an application of parallel minimum circle cover algorithms in [2, 10] runs in $O(\log n)$ time using $O(n)$ processors.

Theorem 5 *Given a convex polygon P with n vertices, we can compute the minimum radius r^* in $O(n \log^2 n)$ time such that two disks of radius r^* with centers at vertices of P cover P.*

Proof: By Theorem 4, our sequential decision algorithm takes $O(n \log n)$ time.

After constructing the segment tree and computing $\beta^+(\mathcal{I}_v)$ and $\beta^-(\mathcal{I}_v)$ for each v in our parallel algorithm, sort the vertices of P and the vertices of $\beta^+(\mathcal{I}_v) \cup \beta^-(\mathcal{I}_v)$ for all nodes v by their x-coordinates. This makes the application of fractional cascading and computation of c_i and d_i

no longer depend on r, by using ranks rather than x-coordinates. So, these steps can be done sequentially.

Since the Cole's improvement [7] again can be applied to the parallel steps (i.e., constructing the segment tree, sorting, and applying parallel circle cover algorithm, that take $O(\log n)$ time using $O(n \log n)$ processors), the time bound can be achieved. **QED**

Our idea for Theorem 5 can be extended to the discrete two-center problem for points when the points are the vertices of a convex polygon.

Theorem 6 *Given a set A of n points that are vertices of a convex polygon, we can compute the minimum radius r^* in $O(n \log^2 n)$ time such that two disks of radius r^* with centers at points of A cover A.*

Proof: Let P be a convex polygon whose vertices are the points of A. Our algorithm performs exactly the same way as the one for Theorem 5, except we compute $\langle\langle a_i, b_i \rangle\rangle$ for each i, where both a_i and b_i are vertices of A. After obtaining a_i and b_i for each i, we check whether there exist j and ℓ such that $A = \langle\langle a_j, b_j \rangle\rangle \cup \langle\langle a_\ell, b_\ell \rangle\rangle$, which is independent of r. **QED**

4 Algorithm for standard two-center problem for a set of points in convex position

Suppose that a set A of n points in convex positions are given. We assume that as input a convex polygon with vertices at the points of A is given, so the points are sorted counterclockwise. We want to find two disks of radius r^* whose union covers A. As usual, the points are numbered $0, \ldots, n-1$ counterclockwise.

For any $i, j \in A = \langle 0, n-1 \rangle$, let r^1_{ij} be the radius of the smallest disk containing $\langle i, j-1 \rangle$ and let r^2_{ij} be the radius of the smallest disk containing $\langle j, i-1 \rangle$. Then, $r^* = \min_{i,j \in A} \max\{r^1_{ij}, r^2_{ij}\}$.

Let $r^*_0 = \min_{j \in A} \max\{r^1_{0,j}, r^2_{0,j}\}$. Let k be the index such that $r^*_0 = \max\{r^1_{0,k}, r^2_{0,k}\}$. Then, A is separated into $\langle 0, k-1 \rangle$ and $\langle k, n-1 \rangle$.

Lemma 9 *For any i, j such that $i, j \in \langle 0, k-1 \rangle$ or $i, j \in \langle k, n-1 \rangle$, $\max\{r^1_{ij}, r^2_{ij}\} > r^*_0$.*

By Lemma 9 we need to consider pairs of i, j such that $i \in \langle 0, k-1 \rangle$ and $j \in \langle k, n-1 \rangle$. We now apply divide-and-conquer with $A_1 = \langle 0, k-1 \rangle$ and $A_2 = \langle k, n-1 \rangle$. (i) Pick the medium-index vertex $\lfloor k/2 \rfloor$ from A_1 and (ii) find the vertex k' such that $\max\{r^1_{\lfloor k/2 \rfloor, k'}, r^2_{\lfloor k/2 \rfloor, k'}\} = \min_{j \in \langle k, n-1 \rangle} \max\{r^1_{\lfloor k/2 \rfloor, j}, r^2_{\lfloor k/2 \rfloor, j}\}$. A_1 is separated into $A_{11} = \langle 0, \lfloor k/2 \rfloor - 1 \rangle$ and $A_{12} = \langle \lfloor k/2 \rfloor, k-1 \rangle$, and A_2 is separated into $A_{21} = \langle k, k'-1 \rangle$ and $A_{22} = \langle k', n-1 \rangle$. By Lemma 9, we consider pairs of points between A_{11} and A_{21}, and between A_{12} and A_{22}. (iii) Repeat recursively (i)–(ii) with $A_1 = A_{11}$ and $A_2 = A_{21}$, and with $A_1 = A_{12}$ and $A_2 = A_{22}$. Recursion stops when A_1 consists of a single point ℓ, and at that time $\min_{j \in A_2} \max\{r^1_{\ell,j}, r^2_{\ell,j}\}$ can be found as in (ii).

Since a smallest disk containing a convex chain can be found in linear time, $\max\{r^1_{ij}, r^2_{ij}\}$ for any i, j can be computed in linear time. Since $r^1_{0,1}, \ldots, r^1_{0,n-1}$ is increasing and $r^2_{0,1}, \ldots, r^2_{0,n-1}$ is decreasing, $\max\{r^1_{0,1}, r^2_{0,1}\}, \ldots, \max\{r^1_{0,n-1}, r^2_{0,n-1}\}$ is decreasing and then increasing. So, r^*

and k can be found in $O(n \log n)$ time by binary search, in which each decision takes $O(n)$ time. Similarly, (ii) takes $O(n \log n)$ time. Since our algorithm invokes at most $O(\log n)$ recursions, and after recursion stops a single point $\ell \in A_1$ requires additional $O(|A_2| \log |A_2|)$ time, we can conclude the following theorem.

Theorem 7 *Given a set A of n points that are vertices of a convex polygon, we can compute the minimum radius r^* in $O(n \log^2 n)$ time such that two disks of radius r^* cover A.*

References

[1] P. K. Agarwal and M. Sharir and E. Welzl, The discrete 2-Center problem, *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pp. 147–155, 1997.

[2] L. Boxer and R. Miller, A parallel circle-cover minimization algorithm, *Inform. Process. Lett.*, 32, pp. 57–60, 1989.

[3] T.M. Chan, More planar two-center algorithms, *Computational Geometry: Theory and Applications*, 13, pp. 189-198, 1999

[4] B. Chazelle, and L.J. Guibas, Fractional cascading: I. A data structuring technique, *Algorithmica*, 1, pp. 133–162, 1986.

[5] B. Chazelle, and L.J. Guibas, Fractional cascading: II. Applications, *Algorithmica*, 1, pp. 163–191, 1986.

[6] S.-W. Cheng, O. Cheong, H. Everett, and R. van Oostrum, Hierarchical vertical decompositions, ray shooting, and circular arc queries in simple polygons, *Proc. 15th Ann. ACM Symp. Comput. Geom.*, 1999.

[7] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM*, 34, pp. 200–208, 1987.

[8] D. Eppstein, Faster construction of planar two-centers, *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, 1997.

[9] J. Friedman, J. Hershberger, and J. Snoeyink, Compliant motion in a simple polygon, *Proc. 5th Ann. ACM Symp. Comput. Geom.,* pages 175–186, 1989.

[10] S. K. Kim, Parallel algorithms for geometric intersection graphs, Ph.D. Thesis, Dept. of Comput. Sci. Eng., U. of Washington, 1990.

[11] C. C. Lee and D. T. Lee, On a circle-cover minimization problem, *Inform. Process. Lett.*, 18, pp. 109–115, 1984.

[12] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM"*, 30, pp. 852–865, 1983.

[13] M. Sharir, A Near-Linear Algorithm for the Planar 2-Center Problem, *Proc. 12th Annu. ACM Sympos. Comput. Geom."*, pp. 106–112, 1996.