Covering Convex Polygons by Two Congruent Disks*

Jongmin Choi† Dahye Jeong† Hee-Kap Ahn‡

Abstract

We consider the planar two-center problem for a convex polygon: given a convex polygon in the plane, find two congruent disks of minimum radius whose union contains the polygon. We present an $O(n \log n)$-time algorithm for the two-center problem for a convex polygon, where n is the number of vertices of the polygon. This improves upon the previous best algorithm for the problem.

1 Introduction

In the problem of covering a region R by a predefined shape Q (such as a disk, a square, a rectangle, a convex polygon, etc.) in the plane, we find k homothets[1] of Q with the same homothety ratio such that their union contains R and the homothety ratio is minimized. This is a fundamental optimization problem [2, 5, 20] arising in analyzing and recognizing shapes, and it has real-world applications in various areas, including computer vision and data mining.

The covering problem has been extensively studied in the context of the *k-center* problem and the *facility location* problem when the region to cover is a set of points and the predefined shape is a disk in the plane. In the last decades, there have been a lot of works for these problems, including exact algorithms for $k = 2$ [4, 13, 14, 15, 35, 37], exact and approximation algorithms for $k > 2$ [2, 20, 22, 25], algorithms in higher dimensional spaces [1, 2, 30], and approximation algorithms for streaming points [3, 8, 12, 24, 27, 39]. There are also works on the k-center problem for small k when the region to cover is a set of disks in the plane, specifically for $k = 1$ [21, 28, 31] and $k = 2$ [9].

In the context of the facility location, there have also been works on the *geodesic k-center* problem for simple polygons [7, 32] and polygonal domains [10], in which we find k points (centers) in the domain in order to minimize the maximum geodesic distance from any point in the domain to its closest center.

In this paper we consider the covering problem for a convex polygon in which we find two congruent disks of minimum radius whose union contains the convex polygon. Thus, our problem can be considered as the *(geodesic) two-center problem for a convex polygon.* See Figure 1 for an illustration.

*This research was supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)) and (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)).

†Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. Email: {icothos,dahyejeong}@postech.ac.kr

‡Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Korea. Email: heekap@postech.ac.kr

[1]For a shape Q in the plane, a (positive) homothet of Q is a set of the form $\lambda Q + v := \{\lambda q + v \mid q \in Q\}$, where $\lambda > 0$ is the homothety ratio, and $v \in \mathbb{R}^2$ is a translation vector.

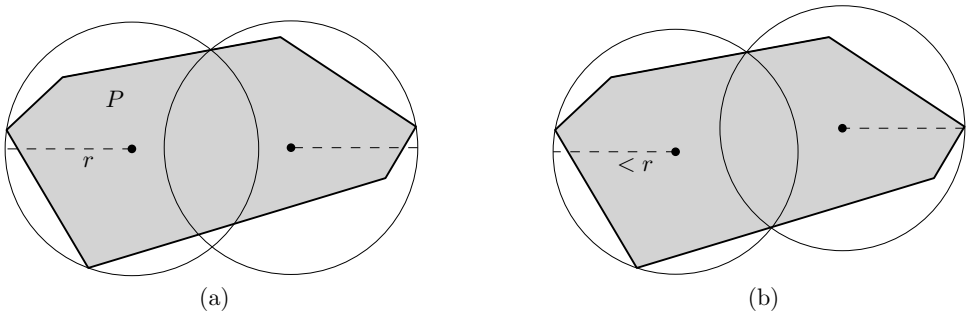

Figure 1: (a) Two congruent disks of radius r whose union covers a convex polygon P. (b) P can be covered by two congruent disks of radius smaller than r.

Previous works. For a convex polygon with n vertices, Shin et al. [36] gave an $O(n^2 \log^3 n)$-time algorithm using parametric search for the two-center problem. They also gave an $O(n \log^3 n)$-time algorithm for the restricted case of the two-center problem in which the centers must lie at polygon vertices. Later, Kim and Shin [26] improved the results and gave an $O(n \log^3 n \log\log n)$-time algorithm for the two-center problem and an $O(n \log^2 n)$-time algorithm for the restricted case of the problem.

There has been a series of work dedicated to variations of the k-center problem for a convex polygon, most of which require certain constraints on the locations of the centers, including the centers restricted to lie on the polygon boundary [33] and on a given polygon edge(s) [17, 33]. For large k, there are quite a few approximation algorithms. For $k \geq 3$, Das et al. [17] gave an $(1+\epsilon)$-approximation algorithm with the centers restricted to lie on the same polygon edge, along with a heuristic algorithm without such restriction. Basappa et al. [11] gave a $(2+\epsilon)$-approximation algorithm for $k \geq 7$ with the centers restricted to lie on the polygon boundary. There is a 2-approximation algorithm for the two-center problem for the convex hull of m points in the plane that supports insertions and deletions of points in $O(\log m)$ time per operation [34].

Our results. We present an $O(n \log n)$-time deterministic algorithm for the two-center problem for a convex polygon P with n vertices. That is, given a convex polygon with n vertices, we can find in $O(n \log n)$ time two congruent disks of minimum radius whose union covers the polygon. This improves upon the $O(n \log^3 n \log\log n)$-time bound of Kim and Shin [26].

Let r^* be the optimal radius value for the two-center problem for P. Our algorithm is twofold. First we solve the sequential decision problem in $O(n)$ time. That is, given a real value r, we can decide whether $r \geq r^*$ in $O(n)$ time. Then we present a parallel algorithm for the decision problem which takes $O(\log n)$ time using $O(n)$ processors, after an $O(n \log n)$-time preprocessing. Using these decision algorithms and applying Cole's parametric search [16], we solve the optimization problem, and return r^* and the two optimal centers of P in $O(n \log n)$ deterministic time.

We observe that if P is covered by two congruent disks D_1 and D_2 of radius r, D_1 covers a connected subchain P_1 of the boundary of P and D_2 covers the remaining subchain P_2 of the boundary of P. Thus, in the sequential decision algorithm, we compute for any point x on the boundary of P, the longest subchain of the boundary of P from x in counterclockwise direction that is covered by a disk of radius r, and the longest subchain of the boundary P from x in clockwise direction that is covered by a disk of radius r. We show that the determinators of the disks defining the two longest subchains change $O(n)$ times while x moves along the boundary of P. We also show that the disks and the longest subchains can be represented by $O(n)$ algebraic functions. Our sequential decision algorithm computes the longest subchains in

$O(n)$ time. Finally, the sequential decision algorithm determines whether there is a point x' on the boundary of P such that the two longest subchains from x', one in counterclockwise direction and one in clockwise direction, cover the polygon boundary in $O(n)$ time.

Our parallel decision algorithm computes the longest subchains in parallel and determines whether there is a point x' on the boundary of P such that the two longest subchains from x', one in counterclockwise and one in clockwise along the boundary of P, cover the polygon boundary in $O(\log n)$ parallel steps using $O(n)$ processors after $O(n \log n)$-time preprocessing. For this purpose, the algorithm finds rough bounds of the longest subchains, by modifying the parallel decision algorithm for the planar two-center problem of points in convex position [15] and applying it for the vertices of P. Then the algorithm computes $O(n)$ algebraic functions of the longest subchains in $O(\log n)$ time using $O(n)$ processors. Finally, it determines in parallel computation whether there is a point x' on the boundary of P such that the two longest subchains from x' covers the polygon boundary.

We can compute the optimal radius value r^* using Cole's parametric search [16]. For a sequential decision algorithm of running time T_S and a parallel decision algorithm of parallel running time T_P using N processors, Cole's parametric search is a technique that computes an optimal value in $O(NT_P + T_S(T_P + \log N))$ time. In our case, $T_S = O(n)$, $T_P = O(\log n)$, and $N = O(n)$. Therefore, we get a deterministic $O(n \log n)$-time algorithm for the two-center problem for a convex polygon with n vertices.

2 Preliminaries

For any two sets X and Y in the plane, we say X *covers* Y if $Y \subseteq X$. We say a set X is r-*coverable* if there is a disk D of radius r covering X. For a compact set A, we use ∂A to denote the boundary of A. We simply say x moves *along* ∂A when x moves in the counterclockwise direction along ∂A. Otherwise, we explicitly mention the direction.

Let P be a convex polygon with n vertices $v_1, v_2, \ldots, v_n$ in counterclockwise order along the boundary of P. Throughout the paper, we assume general circular position on the vertices of P, meaning that no four vertices are cocircular. We denote the subchain of ∂P from a point x to a point y in ∂P in counterclockwise order as $P_{x,y} = \langle x, v_i, v_{i+1}, \ldots, v_j, y \rangle$, where $v_i, v_{i+1}, \ldots, v_j$ are the vertices of P that are contained in the subchain. We call $x, v_i, v_{i+1}, \ldots, v_j, y$ the *vertices* of $P_{x,y}$. By $|P_{x,y}|$, we denote the number of vertices of $P_{x,y}$.

We can define an order on the points of ∂P, with respect to a point $p \in \partial P$. Here, p is considered as a reference point used in determining orders of points along the boundary of P. For two points x and y of ∂P, we use $x <_p y$ if y is farther than x from p in the counterclockwise direction along ∂P. We define $\leq_p, >_p, \geq_p$ accordingly.

For a subchain C of ∂P, we denote by $I_r(C)$ the intersection of the disks of radius r, each centered at a point in C. See Figure 2(a). Observe that any disk of radius r centered at a point $p \in I_r(C)$ covers the entire chain C. Hence, $I_r(C) \neq \emptyset$ if and only if C is r-coverable. The *circular hull* of a set X, denoted by $\alpha_r(X)$, is the intersection of all disks of radius r covering X. See Figure 2(b).

Every vertex of $\alpha_r(C)$ is a vertex of C. The boundary of $\alpha_r(C)$ consists of arcs of radius r, each connecting two vertices of C. The circular hull $\alpha_r(C)$ is dual to the intersection $I_r(C)$, in the sense that every arc of $\alpha_r(C)$ is on the circle of radius r centered at a vertex of $I_r(C)$, and every arc of $I_r(C)$ is on the circle of radius r centered at a vertex of $\alpha_r(C)$. This implies that $\alpha_r(C) \neq \emptyset$ if and only if $I_r(C) \neq \emptyset$. Therefore, $\alpha_r(C)$ is nonempty if and only if C is r-coverable.

Let S be the set of vertices of a subchain C of ∂P. If a disk covers C, it also covers S. If a disk covers S, it also covers C since it covers every line segment connecting two points in S,

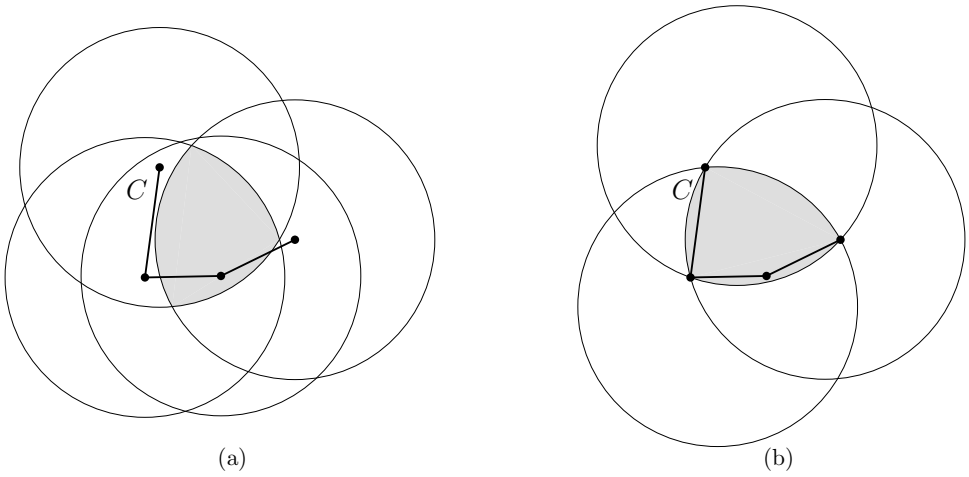

Figure 2: C is a subchain of ∂P and S is the vertex set of C. (a) The sets $I_r(C)$ and $I_r(S)$ are the same. (b) The sets $\alpha_r(C)$ and $\alpha_r(S)$ are the same.

due to the convexity of a disk. Therefore, $\alpha_r(C)$ and $\alpha_r(S)$ are the same, and $I_r(C)$ and $I_r(S)$ are the same. See Figure 2(b).

For a vertex v of $\alpha_r(C)$, we denote by $\mathsf{ccw}(v)$ its counterclockwise neighbor on $\partial\alpha_r(C)$, and by $\mathsf{cw}(v)$ its clockwise neighbor on $\partial\alpha_r(C)$. We denote by $\gamma(v)$ the arc of $\alpha_r(C)$ connecting v and $\mathsf{ccw}(v)$ of $\alpha_r(C)$. By $\delta(v)$, we denote the supporting disk of the arc $\gamma(v)$ of $\alpha_r(C)$, that is, the disk containing $\gamma(v)$ in its boundary. We may use $\alpha(C)$ and $I(C)$ to denote $\alpha_r(C)$ and $I_r(C)$, respectively, if it is understood from context. Since $\alpha(C)$ and $\alpha(S)$ are the same, we obtain the following observation on subchains from the observations on planar points [18, 23].

Observation 1 ([18, 23])**.** *For a subchain C of ∂P, the followings hold.*

1. *For any subchain $C' \subseteq C$, $\alpha_r(C') \subseteq \alpha_r(C)$.*
2. *A vertex of C appears as a vertex in $\alpha_r(C)$ if and only if C is r-coverable by a disk containing the vertex on its boundary.*
3. *An arc of radius r connecting two vertices of C appears as an arc of $\alpha_r(C)$ if and only if C is r-coverable by the supporting disk of the arc.*

For a point $x \in \partial P$, let $f_r(x)$ be the farthest point on ∂P from x in the counterclockwise direction along ∂P such that $P_{x,f_r(x)}$ is r-coverable. We denote by $D_1^r(x)$ the disk of radius r that covers $P_{x,f_r(x)}$. Similarly, let $g_r(x)$ be the farthest point on ∂P from x in the clockwise direction such that $P_{g_r(x),x}$ is r-coverable, and denote by $D_2^r(x)$ the disk of radius r that covers $P_{g_r(x),x}$. Note that x may not lie on the boundaries of D_1^r and D_2^r. We may use $f(x)$, $D_1(x)$, $g(x)$, and $D_2(x)$ by omitting the subscript and superscript r in the notations, if they are understood from context. See Figure 3 for illustrations of $f(x)$ and $D_1(x)$.

Since we can determine in $O(n)$ time whether P is r-coverable [30], we assume that P is not r-coverable in the remainder of the paper. For a fixed r, consider any two points t and t' in ∂P satisfying $t <_t t' <_t f(t)$. Then $P_{t',f(t)}$ is r-coverable, which implies $f(t) \leq_{t'} f(t')$. Thus, we have the following observation.

Observation 2. *For a fixed r, as x moves along ∂P in the counterclockwise direction, both $f(x)$ and $g(x)$ move monotonically along ∂P in the counterclockwise direction.*

3 Sequential Decision Algorithm

In this section, we consider the decision problem: given a real value r, decide whether $r \geq r^*$, that is, whether there are two congruent disks of radius r whose union covers P.

For a point x moving along ∂P, we consider two functions, $f(x)$ and $g(x)$. If there is a point $x \in \partial P$ such that $f(x) \geq_x g(x)$, the union of $P_{x,f(x)}$ and $P_{g(x),x}$ is ∂P. Thus there are two congruent disks of radius r whose union covers P, and the decision algorithm returns `yes`. Otherwise, we conclude that $r < r^*$, and the decision algorithm returns `no`. For a subchain $P_{x,y}$ of ∂P, we use $\alpha(x, y)$ to denote $\alpha(P_{x,y})$, and $I(x, y)$ to denote $I(P_{x,y})$.

3.1 Characterizations

For a fixed r, $I(x, f(x))$ is a point and $\alpha(x, f(x))$ is a disk such that $I(x, f(x))$ is the center of $\alpha(x, f(x))$. Moreover, $\alpha(x, f(x))$ and $D_1(x)$ are the same. Observe that $D_1(x)$ is determined by two or three vertices of $P_{x,f(x)}$, which we call the *determinators* of $D_1(x)$. For our purpose, we define four *types* of $D_1(x)$ by its determinators: (`T1`) x, $f(x)$, and one vertex. (`T2`) x and $f(x)$. (`T3`) $f(x)$ and one vertex. (`T4`) $f(x)$ and two vertices. See Figure 3 for an illustration of the four types.

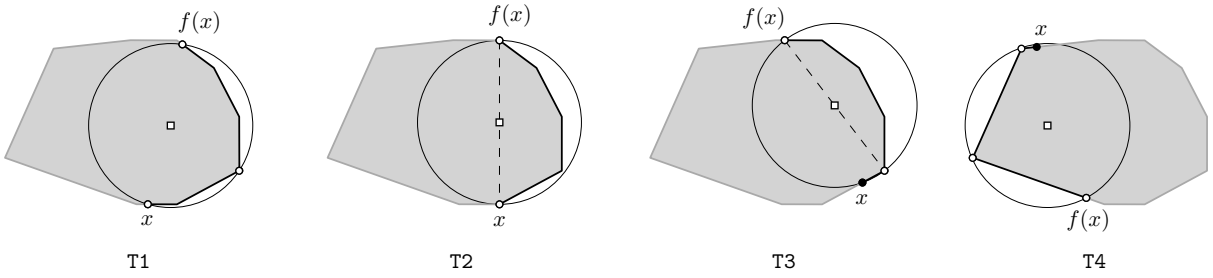

Figure 3: Four types of $D_1(x)$ and its determinators (small circles).

We denote by $e(a)$ the edge of P containing a point $a \in \partial P$. If a is a vertex of P, $e(a)$ denotes the edge of P incident to a lying in the counterclockwise direction from a. For a point x moving along ∂P, the *combinatorial structure* of $f(x)$ is defined to be the set consisting of $e(x)$, $e(f(x))$, and the determinators of $D_1(x)$. We call each point x on ∂P at which the combinatorial structure of $f(x)$ changes a *breakpoint* of $f(x)$. For $x \in \partial P$ lying in between two consecutive breakpoints, we can compute $f(x)$ using $e(x)$, $e(f(x))$, and $D_1(x)$ in $O(1)$ time.

Consider x moving along ∂P starting from x_0 on ∂P in counterclockwise direction. Let $x_1 = f(x_0)$, $x_2 = f(x_1)$ and $x_3 = f(x_2)$. We simply use the index i instead of x_i for $i = 0, \ldots, 3$ if it is understood from context. For instance, we use $P_{i,j}$ to denote P_{x_i,x_j}, and $\leq_i$ to denote $\leq_{x_i}$. For the rest of the section, we describe how to handle the case that x moves along $P_{0,1}$. The cases that x moves along $P_{1,2}$ and $P_{2,3}$ can be handled analogously. As x moves along $P_{0,1}$, $f(x)$ moves along $P_{1,2}$ in the same direction by Observation 2.

Lemma 1. *For any fixed $r \geq r^*$, the union of $P_{0,1}$, $P_{1,2}$, and $P_{2,3}$ is ∂P.*

Proof. If P is r-coverable, $P_{0,1}$ is ∂P. Assume that P is not r-coverable. For any fixed $r \geq r^*$, there are two congruent disks D_1 and D_2 of radius r whose union covers P. Let y and z be the points of ∂P such that $P_{y,z}$ is covered by D_1, and $P_{z,y}$ is covered by D_2. Without loss of generality, assume $x_0 \in P_{y,z}$. Then $z \leq_0 f(y) \leq_0 x_1$ along ∂P, because $P_{y,z}$ is covered by D_1 and by Observation 2. See Figure 4 for an illustration. If $x_1 \leq_0 y$, then $y \leq_1 x_2$ and thus $x_0 \leq_y x_3$ by Observation 2. If $y \leq_0 x_1$, then $x_0 \leq_y x_2$. Thus, the union of $P_{0,1}$, $P_{1,2}$, and $P_{2,3}$ is

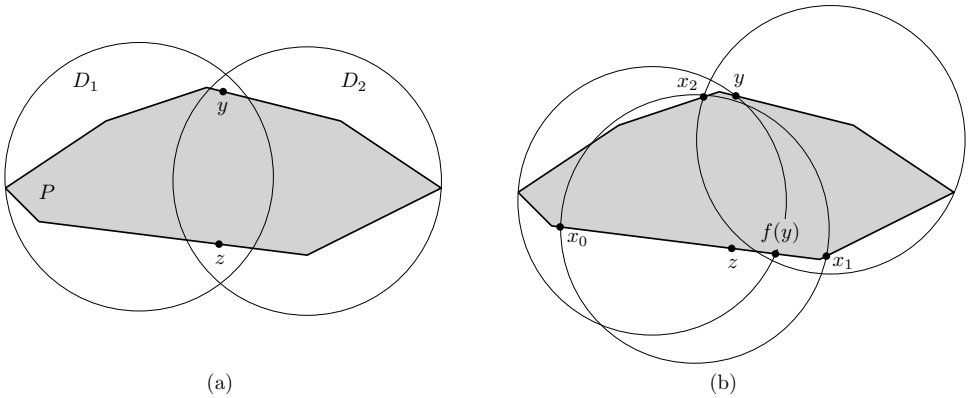

Figure 4: (a) $P_{y,z}$ is covered by D_1 and $P_{z,y}$ is covered by D_2. (b) If $x_0 \in P_{y,z}$, then $z \leq_0 f(y) \leq_0 x_1$ and $y \leq_1 x_2$.

∂P. □

The structure of a circular hull can be expressed by the circular sequence of arcs appearing on the boundary of the circular hull. There is a 1-to-1 correspondence between the set of breakpoints of $f(x)$ for x moving along $P_{0,1}$ and the set of structural changes to $\alpha(x, f(x))$, because $D_1(x)$ and $\alpha(x, f(x))$ are the same. Thus, we maintain $D_1(x)$ for x moving along $P_{0,1}$ and capture every structural change to $\alpha(x, f(x))$. Observe that the boundary of $\alpha(x, f(x))$ consists of a connected boundary part of $\alpha(x, x_1)$, a connected boundary part of $\alpha(x_1, f(x))$, and two arcs of $D_1(x)$ connecting $\alpha(x, x_1)$ and $\alpha(x_1, f(x))$. See Figure 5 for an illustration.

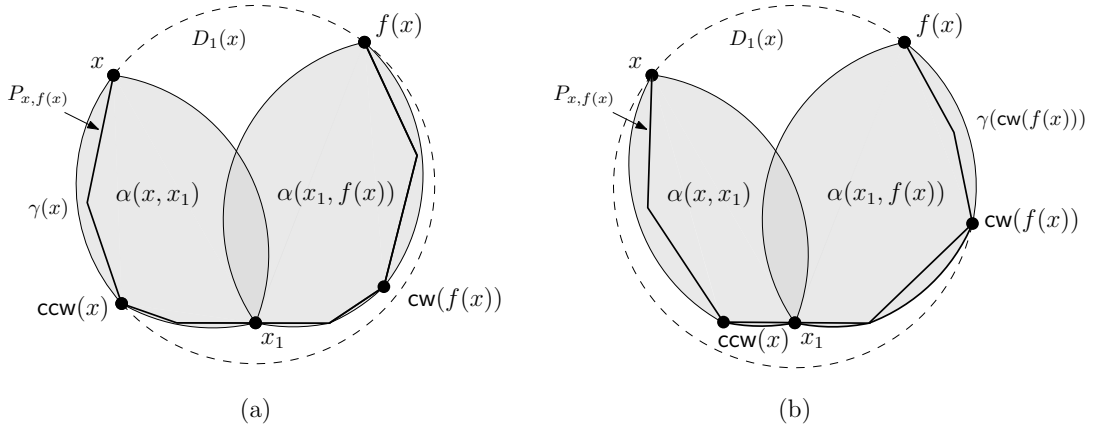

Figure 5: $D_1(x)$ of type T1 has three determinators which are x, $f(x)$, and a vertex v of $P_{x,f(x)}$. (a) If v is on the boundary of $\alpha(x, x_1)$, $D_1(x)$ is $\delta(x)$ of $\alpha(x, x_1)$. (b) If v is on the boundary of $\alpha(x_1, f(x))$, $D_1(x)$ is $\delta(\mathsf{cw}(f(x)))$.

The following lemma gives some characterizations to the four types of $D_1(x)$. Recall that for a vertex v, $\delta(v)$ is the supporting disk of the arc $\gamma(v)$ of a circular hull, that is, $\delta(v)$ is the disk containing $\gamma(v)$ on its boundary.

Lemma 2. *For a point x on the boundary of P, we have the followings.*

(A) If $D_1(x)$ is of type T1, it is $\delta(x)$ of $\alpha(x, x_1)$ or $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$.

(B) If $D_1(x)$ is of type T2, x and $f(x)$ are at Euclidean distance $2r$.

(C) If $D_1(x)$ is of type T3 or T4 and it has x on its boundary, $D_1(x)$ is $\delta(x)$ of $\alpha(x, x_1)$ or $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$. Moreover, for any point y in the interior of $P_{x,u}$, $D_1(y)$ and $D_1(x)$ are of the same type, where u is the determinator of $D_1(x)$ closest to x in counterclockwise order.

Proof. For $D_1(x)$ of type T1, it has three determinators x, $f(x)$, and a vertex v of $P_{x,f(x)}$ lying on the boundary of $\alpha(x, f(x))$. Assume that v is on the boundary of $\alpha(x, x_1)$. Then the boundary portion of $\alpha(x, f(x))$, from x to v in counterclockwise order, is from the boundary of $\alpha(x, x_1)$, and $\mathsf{ccw}(x)$ of $\alpha(x, f(x))$ lies on the boundary of $\alpha(x, x_1)$. Since $\gamma(x)$ of $\alpha(x, x_1)$ is on the boundary of $D_1(x)$, $\delta(x)$ of $\alpha(x, x_1)$ is $D_1(x)$. See Figure 5(a). A similar argument can be made for the case that v is on the boundary of $\alpha(x_1, f(x))$. See Figure 5(b). Therefore, $D_1(x)$ is $\delta(x)$ of $\alpha(x, x_1)$ or $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$.

For $D_1(x)$ of type T2, there are exactly two determinators, x and $f(x)$, and their Euclidean distance is $2r$.

For $D_1(x)$ of type T3 or T4 that has x on its boundary, we can show that $D_1(x)$ is $\delta(x)$ of $\alpha(x, x_1)$ or $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$, by an argument similar to that for $D_1(x)$ of type T1. By Observation 2, we have $f(x) \leq_u f(y)$ for any point y in the interior of $P_{x,u}$. Thus the determinators of $D_1(x)$ lying in between u and $f(x)$ are all contained in $P_{y,f(y)}$. Therefore, $D_1(x)$ and $D_1(y)$ are of the same type. □

The function $f(x)$ can be computed using $e(x)$, $e(f(x))$, and the disk $D_1(x)$. By Lemma 2, if $D_1(x)$ is of type T1, T3 or T4, $D_1(x)$ is either $\delta(x)$ of $\alpha(x, x_1)$ or $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$. The disk $\delta(x)$ of $\alpha(x, x_1)$ can be computed using x and $\mathsf{ccw}(x)$ of $\alpha(x_1, f(x))$ and the disk $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$ can be computed using $f(x)$ and $\mathsf{cw}(f(x))$ of $\alpha(x_1, f(x))$. Thus, our decision algorithm computes $f(x)$ by maintaining $e(x)$, $e(f(x))$, $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ and $\mathsf{cw}(f(x))$ of $\alpha(x_1, f(x))$.

Therefore, we compute the changes to $e(x)$ and $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ for a point x moving along $P_{0,1}$, and compute the changes to $e(y)$ and $\mathsf{cw}(y)$ of $\alpha(x_1, y)$ for a point y moving along $P_{1,2}$. We call the points inducing these changes the *event points*. From this, we detect the combinatorial changes to $f(x)$.

3.2 Data structures and decision algorithm

In this section, we focus on the case that x moves along $P_{0,1}$ and $f(x)$ moves along $P_{1,2}$. For the cases where x moves along $P_{1,2}$ or $P_{2,3}$, we can apply the same algorithm. For ease of description, we use P_L to denote $P_{0,1}$, and P_R to denote $P_{1,2}$. Let E_L be the set of event points such that $e(x)$ or $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ changes for a point x moving along P_L, and let E_R be the set of event points such that $e(y)$ or $\mathsf{cw}(y)$ of $\alpha(x_1, y)$ changes for a point y moving along P_R.

To compute E_L and E_R, we use the semi-dynamic (insertion-only) data structure that was given for the two-center problem for points in the plane [38]. The structure maintains the circular hull for points in the plane that are inserted in increasing order of their x-coordinates or in the order sorted around a point. Since the vertices of P are already sorted around any fixed point in the interior of P, we can use the data structure for our purpose.

Lemma 3 (Theorem 5 in [38])**.** *We can maintain the circular hull of a set Q of points such that if a new point to the right of all points of Q is inserted, in $O(1)$ amortized time we can decide whether $\alpha(Q)$ exists, and if yes, update $\alpha(Q)$.*

We can modify the algorithm to work not only for point insertions, but also for edge insertions in order along a chain. Let $v_1, \ldots, v_i$ be the vertices of P inserted so far in order from v_1. When

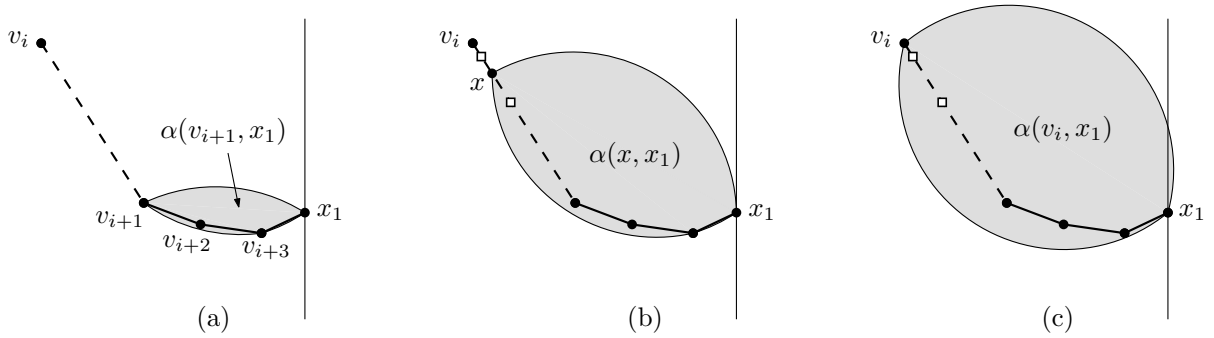

Figure 6: (a) The vertices of $\alpha(v_{i+1}, x_1)$ consists of v_{i+1}, v_{i+3} and x_1. (b) While x moves along the edge v_iv_{i+1} from v_{i+1} to v_i, x appears on $\alpha(x, x_1)$. During this process, v_{i+1} and v_{i+3} are repeatedly popped from the stack. The empty squares represent the intersection of the edge v_iv_{i+1} and the supporting circle of the arc connecting the popped vertex and the vertex at the top of the stack at the moment. (c) The circular hull $\alpha(v_i, x_1)$.

v_{i+1} is inserted, we compute the points z on edge v_iv_{i+1} at which a structural change to $\alpha(v_1, z)$ occurs.

Lemma 4. *The number of event points in E_L is $O(|P_L|)$. We can compute E_L in $O(|P_L|)$ time.*

Proof. Imagine that x moves along P_L in *clockwise* order, from x_1 to x_0. Then $e(x)$ changes $O(|P_L|)$ times at the vertices of P_L.

Consider the changes to $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ occurred by the circular hull of the the vertices that are inserted one by one from $P_L = \langle x_0, \ldots, v_i, v_{i+1}, \ldots, x_1\rangle$, in reverse order from x_1. We store the vertices of the circular hull in a stack in the order. When v_i is inserted, we already have $\alpha(v_{i+1}, x_1)$. We also have the vertices of the circular hull stored in the stack in clockwise order with v_{i+1} at the top. Our goal is to compute $\alpha(v_i, x_1)$ and the points x on edge v_iv_{i+1} at which $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ changes. To do this, we pop vertices repeatedly from the stack until $\mathsf{ccw}(v_i)$ of $\alpha(v_i, x_1)$ becomes the top element of the stack. When a vertex v is popped from the stack, we consider the supporting disk D of the arc connecting v and the vertex at the top of the stack at the moment, and compute the intersection of ∂D with v_iv_{i+1}. Since $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ changes when x reaches such an intersection, we consider those intersection points x as the event points inducing the changes to $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$. See Figure 6 for an illustration of this process.

Observe that once a vertex is popped from the stack, it is never inserted to the stack again. Therefore, $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ changes $O(|P_L|)$ times while x moves along P_L, and we can compute the event points in $O(|P_L|)$ time. □

From Lemma 4, we obtain the following Corollary.

Corollary 1. *For a point y moving along P_R, $e(y)$ and $\mathsf{cw}(y)$ of $\alpha(x_1, y)$ change $O(|P_R|)$ times. We can compute the event points y at which $\mathsf{cw}(y)$ of $\alpha(x_1, y)$ changes in $O(|P_R|)$ time.*

Let E' be the set of points x on P_L such that x is an event point in E_L or $f(x)$ is an event point in E_R. Let $\tilde{P}$ be the set of the pieces in the subdivision of P_L induced by E'. There are $O(|P_L| + |P_R|)$ pieces in $\tilde{P}$ such that each piece is a segment and any point x in a piece has the same $e(x)$, $e(f(x))$, $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ and $\mathsf{cw}(f(x))$ of $\alpha(x_1, f(x))$.

Lemma 5. *For any fixed $r \geq r^*$, there are $O(n)$ breakpoints of $f(x)$ and $g(x)$.*

Proof. For a point x moving along P_L, $e(x)$ and $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ change $O(|P_L|)$ times by Lemma 4. Since $f(x)$ moves along P_R while x moves along P_L, $e(f(x))$ and $\mathsf{cw}(f(x))$ of

$\alpha(x_1, f(x))$ change $O(|P_R|)$ times by Corollary 1. Combining them, there are $O(|P_L| + |P_R|)$ changes to $e(x)$, $e(f(x))$, $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$, and $\mathsf{cw}(f(x))$ of $\alpha(x_1, f(x))$ while x moves along P_L.

Let T be a segment in $\tilde{P}$. We count the breakpoints of $f(x)$ in the interior of T induced by the following three cases: (1) the type of $D_1(x)$ changes to `T3` or `T4`, (2) the determinators of $D_1(x)$ changes while $D_1(x)$ remains to be of type `T1`, and (3) the type of $D_1(x)$ switches between `T1` and `T2`.

Consider the breakpoints induced by case (1). Once the type of $D_1(x)$ changes to `T3` or `T4`, the determinators remain the same while x moves along T by Lemma 2. Thus, there is at most one breakpoint induced by case (1).

A breakpoint induced by case (2) occurs at $x = t$ where $\delta(t)$ of $\alpha(t, x_1)$ and $\delta(\mathsf{cw}(f(t)))$ of $\alpha(x_1, f(t))$ are the same by Lemma 2. For $\mathsf{ccw}(x)$ of $\alpha(x, x_1)$ and $\mathsf{cw}(f(x))$ of $\alpha(x_1, f(x))$, there is at most one disk such that $\delta(t)$ of $\alpha(t, x_1)$ and $\delta(\mathsf{cw}(f(t)))$ of $\alpha(x_1, f(t))$ are the same. Thus, there is at most one breakpoint induced by case (2).

A breakpoint induced by case (3) occurs at $x = t$ where $f(t)$ is at Euclidean distance $2r$ from t. Assume that $D_1(x)$ is $\delta(x)$ of $\alpha(x, x_1)$. Then there are at most two such moments t with $\mathsf{ccw}(t)$ on $\partial D_1(t)$ at which the Euclidean distance between t and $f(t)$ is $2r$. For the case that $D_1(x)$ is $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$, there are at most two such moments by a similar argument. Thus, there are $O(1)$ breakpoints induced by case (3).

Since there are $O(|P_L| + |P_R|)$ segments in $\tilde{P}$ and each segment has $O(1)$ breakpoints in its interior, there are $O(|P_L| + |P_R|)$ breakpoints of $f(x)$ for x moving along P_L. By a similar argument, we can conclude that there are $O(|P_L| + |P_R| + |P_{2,3}|) = O(n)$ breakpoints of $f(x)$ for x moving along P_L, P_R, and $P_{2,3}$. We can also show that there are $O(n)$ breakpoints of $g(x)$ by a similar argument. □

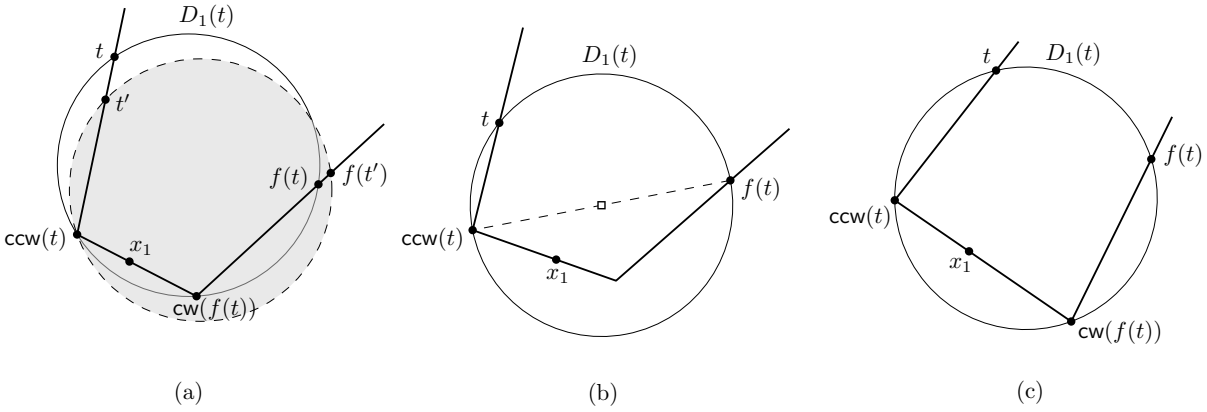

Figure 7: Changes to the determinators of $D_1(x)$. (a) At $x = t$, the determinators of $D_1(x)$ of type `T1` change from $\{x, \mathsf{cw}(f(x)), f(x)\}$ to $\{x, \mathsf{ccw}(x), f(x)\}$. (b) At $x = t$, the type of $D_1(x)$ changes to `T3`. (c) At $x = t$, the type of $D_1(x)$ changes to `T4`.

Using Lemmas 2, 3, 4, and 5, we compute the breakpoints of $f(x)$ by moving x along ∂P from x_0.

Lemma 6. *For a fixed $r \geq r^*$, the breakpoints of $f(x)$ and $g(x)$ can be computed in $O(n)$ time.*

Proof. We compute the segments of $\tilde{P}$ in $O(|P_L|+|P_R|)$ time, and then compute the breakpoints of $f(x)$ for points $x \in P_L$.

We compute $D_1(x_0)$ and its determinators. To this end, we find the edge vv' of P containing $f(x_0)$ in $O(n)$ time using Lemma 3. Then in $O(n)$ time, we compute $f(x_0)$ and the type of $D_1(x_0)$ as follows.

- If $D_1(x_0)$ is of type `T1` or `T2`, $f(x_0)$ is the event point at which $\alpha(x_0, f(x_0))$ becomes a disk of radius r, and thus we can find $D_1(x_0)$ and its determinators in $O(n)$ time.
- If $D_1(x_0)$ is of type `T3` or `T4`, x_0 may not lie on $\partial D_1(x_0)$. If $D_1(x_0)$ is of type `T3`, we find the point z on vv' for every vertex u of $\alpha(x_0, v)$ such that the Euclidean distance between z and u is $2r$. In this way, we find $f(x_0)$ in $O(n)$ time for the case. If $D_1(x_0)$ is of type `T4`, we find $f(x_0)$ in $O(n)$ time by computing the intersection of vv' and the supporting disk of every arc of $\alpha(x_0, v)$.

We then continue to compute $f(x)$ for each type of $D_1(x)$ by checking the breakpoints of $f(x)$ for x moving along P_L starting from x_0. Imagine that x moves along a segment T in P_L between two event points of E_L consecutive along P_L, and $f(x)$ moves along a segment T' in P_R between two event points of E_R consecutive along P_R. If the type of $D_1(x)$ changes to `T1`, `T3`, or `T4`, $D_1(x)$ becomes $\delta(x)$ of $\alpha(x, x_1)$ or $\delta(\mathsf{cw}(f(x)))$ of $\alpha(x_1, f(x))$ by Lemma 2. At that moment, the determinators of $D_1(x)$ must lie on the boundary of $\alpha(x, f(x))$. By the general circular position assumption, there can be at most three polygon vertices lying on the boundary of $D_1(x)$. Therefore, the determinators of $D_1(x)$ are at most three elements among the vertices $\mathsf{ccw}(x)$, $\mathsf{ccw}(\mathsf{ccw}(x))$, $\mathsf{ccw}(\mathsf{ccw}(\mathsf{ccw}(x)))$ of $\alpha(x, x_1)$, $\mathsf{cw}(f(x))$, $\mathsf{cw}(\mathsf{cw}(f(x)))$, and $\mathsf{cw}(\mathsf{cw}(\mathsf{cw}(f(x))))$ of $\alpha(x_1, f(x))$. Since there are $O(n)$ changes to these vertices by an argument similar to Lemma 4, we can compute $f(x)$ and $D_1(x)$ under the assumption that $D_1(x)$ is of type `T1`, `T3`, and `T4` in $O(n)$ time. See Figure 7 for an illustration. If $D_1(x)$ is of type `T2`, then the Euclidean distance between x and $f(x)$ is $2r$ by Lemma 2.

By comparing two $f(x)$ functions, one for the current type of $D_1(x)$ and one for other types of $D_1(x)$, we can compute in $O(1)$ time the next breakpoint, the new determinators, and the type of $D_1(x)$ for x right after the breakpoint. Since there are $O(n)$ breakpoints by Lemma 5, all breakpoints of $f(x)$ can be computed in $O(n)$ time. Similarly, the breakpoints of $g(x)$ can be computed in $O(n)$ time. □

Recall that our algorithm returns `yes` if there exists a point $x \in \partial P$ such that $f(x) \geq_x g(x)$. Otherwise it returns `no`. Hence, using Lemma 6, we have the following theorem.

Theorem 1. *Given a convex polygon P with n vertices in the plane and a radius r, we can decide whether there are two congruent disks of radius r covering P in $O(n)$ time.*

4 Parallel Decision Algorithm

Given a real value r, our parallel decision algorithm computes $f(x)$ and $g(x)$ that define the two longest subchains of ∂P from x, one in counterclockwise direction and one in clockwise direction, respectively, that can be covered by a disk of radius r containing x in parallel. It also determines whether there is a point $x \in \partial P$ such that $f(x) \geq_x g(x)$, in parallel. To do this efficiently, our algorithm first finds rough bounds of $f(x)$ and $g(x)$ by modifying the parallel decision algorithm for the two-center problem for points in convex position by Choi and Ahn [15] and applying it for the vertices of P. Then our algorithm computes $f(x)$ and $g(x)$ exactly.

The parallel decision algorithm by Choi and Ahn runs in two phases: the preprocessing phase and the decision phase. In the preprocessing phase, their algorithm runs sequentially without knowing r. It partitions the point set into subsets of equal size along the convex hull of the

point set and finds two subsets containing the two points that determine an optimal partition in $O(n \log n)$ time. In the decision phase, their algorithm runs in parallel for a given value r. It constructs a data structure that supports intersection queries of a subset of disks centered at input points in $O(\log n)$ parallel time using $O(n)$ processors, after $O(n \log n)$-time preprocessing.

In our problem, the optimal two disks must cover the edges of P as well as the vertices of P. Thus we modify the preprocessing phase of the algorithm by Choi and Ahn as follows. Their algorithm partitions the vertices of P into two subsets $S_1 = \{v_1, \ldots, v_k\}$ and $S_2 = \{v_{k+1}, \ldots, v_n\}$, each consisting of consecutive vertices along ∂P such that there are $v_i \in S_1$ and $v_j \in S_2$ satisfying $\{v_i, v_{i+1}, \ldots, v_{j-1}\} \subset D_1$ and $\{v_j, v_{j+1}, \ldots, v_{i-1}\} \subset D_2$ for an optimal pair (D_1, D_2) of disks for the vertices of P. The indices of vertices are cyclic such that $n + k \equiv k$ for any integer k. Then in $O(n \log n)$ time, it finds $O(n/\log^6 n)$ pairs of subsets, each consisting of $O(\log^6 n)$ consecutive vertices such that there is one pair (U, W) of sets with $v_i \in U$ and $v_j \in W$, where v_i and v_j are the vertices that determine the optimal partition. Our algorithm, in the preprocessing phase, partitions ∂P into two subchains. Then, it partitions ∂P into $O(n/\log^6 n)$ subchains, each consisting of $O(\log^6 n)$ consecutive vertices, and computes $O(n/\log^6 n)$ pairs of the subchains such that at least one pair has x in one subchain and x' in the other subchain, and $P_{x,x'}$ and $P_{x',x}$ is r^*-coverable.

We also modify the decision phase of the algorithm by Choi and Ahn as follows. Their algorithm constructs a data structure in $O(\log n)$ parallel time using $O(n)$ processors, that for a query with r computes $I_r(u, w)$, where $u \in U', w \in W'$ for any pair (U', W') among the $O(n/\log^6 n)$ pairs. Then it computes $I(u, w)$ in $O(\log n)$ time and determines if $I(u, w) = \emptyset$ in $O(\log^3 \log n)$ time using the data structure. Our algorithm, in the decision phase, constructs a data structure that for a query with r computes $I_r(v_i, v_j)$ and $I_r(v_j, v_i)$ for $v_i \in P_1$, $v_j \in P_2$, where (P_1, P_2) is one of the $O(n/\log^6 n)$ pairs of subchains computed in our preprocessing phase. Our data structure also determines if $I(v_i, v_j) = \emptyset$.

Using the data structure, our algorithm gets rough bounds of $f(x)$ and $g(x)$. Then it computes $f(x)$ and $g(x)$ exactly. In doing so, it computes all breakpoints of $f(x)$ and $g(x)$, and their corresponding combinatorial structures, and determines whether there exists $x \in \partial P$ such that $f(x) \geq_x g(x)$.

4.1 Preprocessing phase

For ease of description, we use $f^*(x)$ to denote $f_{r^*}(x)$, and $g^*(x)$ to denote $g_{r^*}(x)$. Our algorithm partitions ∂P into two subchains such that $P_{x,x'}$ and $P_{x',x}$ are r^*-coverable, for x and x' contained in each subchain. Then it computes $O(n/\log^6 n)$ pairs of subchains of ∂P, each consisting of $O(\log^6 n)$ consecutive vertices.

To this end, our algorithm computes a step function $F(x)$ approximating $f^*(x)$ and a step function $G(x)$ approximating $g^*(x)$ on the same set of intervals of the same length. More precisely, at every $(\log^6 n)$-th vertex v from v_1 along ∂P, it evaluates step functions $F(v)$ and $G(v)$ on r^* such that $f^*(v) \leq_v F(v)$ and $g^*(v) \geq_v G(v)$. See Figure 8(a). In each interval, the region bounded by $F(x)$ from above and by $G(x)$ from below is a rectangular cell. Thus, there is a sequence of $O(n/\log^6 n)$ rectangular cells of width at most $\log^6 n$. See Figure 8(b). Observe that every intersection of $f^*(x)$ and $g^*(x)$ is contained in one of the rectangular cells. Thus, we focus on the sequence of rectangular cells bounded in between $F(x)$ and $G(x)$, which we call the *region of interest* (ROI shortly). In addition, we require $F(x)$ and $G(x)$ to approximate $f^*(x)$ and $g^*(x)$ tight enough such that each rectangular cell can be partitioned further by horizontal lines into disjoint rectangular cells of height at most $\log^6 n$, and in total there are $O(n/\log^6 n)$ disjoint rectangular cells of width and height at most $\log^6 n$ in ROI. See Figure 8(c).

In Lemmas 7 and 8, we show how to partition ∂P into two subchains. For any two points

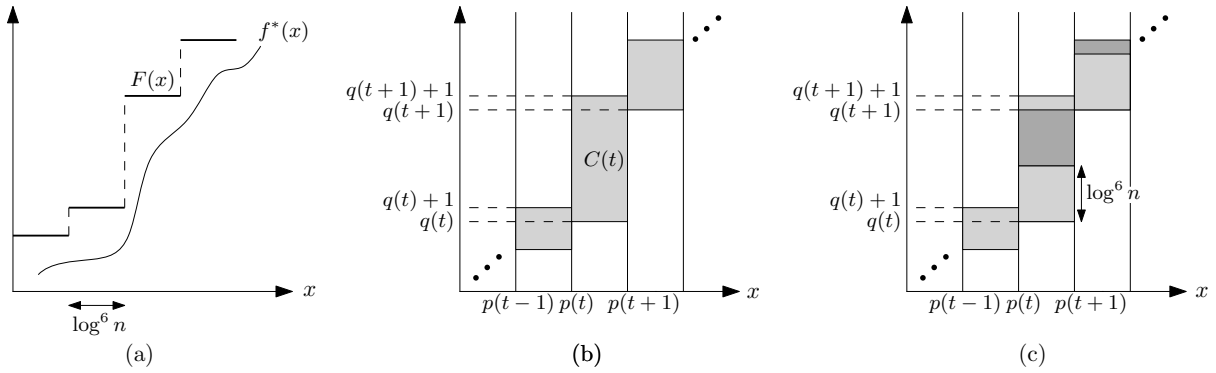

Figure 8: The scale of the axes are labeled with the indices of vertices. (a) Step function $F(x)$ satisfying $f^*(x) \leq F(x)$, with intervals, each consisting of $\log^6 n$ consecutive vertices. (b) Cell $C(t) := [p(t), p(t+1)] \times [q(t), q(t+1)+1]$. The width of each cell is at most $\log^6 n$. (c) $C(t)$ is partitioned further into three subcells, each corresponding to a group consisting of at most $\log^6 n$ consecutive indices in $[q(t), q(t+1)+1]$.

$x, y \in \partial P$, let $\tau(x, y)$ be the smallest value such that $P_{x,y}$ is $\tau(x, y)$-coverable.

Note that $\tau(x, y)$ is a continuous function while x or y moves along the polygon boundary. For a point $p \in \partial P$, let $h(p)$ be the farthest point from p in counterclockwise direction along ∂P that satisfies $\tau(p, h(p)) \leq \tau(h(p), p)$.

Lemma 7. *Given a point $p \in \partial P$, we can find $h(p)$ in $O(n \log n)$ time.*

Proof. For any two vertices u and v of P, we can compute $\tau(u, v)$ in $O(n)$ time using the algorithm for computing the minimum enclosing disk [29]. Observe that, $\tau(u, x)$ increases and $\tau(x, u)$ decreases while x moves from u in counterclockwise direction along ∂P. Thus, given a point $p \in \partial P$, we can find the edge $v_i v_{i+1}$ that contains $h(p)$ using binary search in $O(n \log n)$ time.

To compute $\tau(p, q)$ for any two points $p, q \in \partial P$, consider the intersection $E(p, q)$ of cones, each cone expressed as $(x - u_x)^2 + (y - u_y)^2 \leq z^2$ for a vertex $u = (u_x, u_y)$ of $P_{p,q}$ in the xyz-coordinate system of $\mathbb{R}^3$. Since the intersection of cone $(x-u_x)^2+(y-u_y)^2 \leq z^2$ and $z = r$ is the disk of radius r centered at (u_x, u_y) contained in plane $z = r$, $E(p, q)$ represents $I_r(p, q)$ for all r. Thus, $\tau(p, q)$ is the minimum z-coordinate of the points in $E(p, q)$. The intersection of n translates of a cone with apices in convex position in the plane has complexity $O(n)$, and it can be computed in $O(n)$ time [6]. While q moves along $v_i v_{i+1}$, $\tau(p, q)$ can be obtained from the intersection $E(p, v_i)$ and $E(v_i, q)$. Thus $\tau(q, p)$ has complexity $O(n)$ for all $q \in v_i v_{i+1}$, and it can be computed in $O(n)$ time. We can find $h(p)$ by comparing $\tau(p, q)$ and $\tau(q, p)$ for all $q \in v_i v_{i+1}$. Since the binary search for finding the edge containing $h(p)$ dominates the running time, it takes $O(n \log n)$ time in total. □

Lemma 8. *$P_{v_1,h(v_1)}$ and $P_{h(v_1),v_1}$ form a partition of ∂P such that there are $x \in P_{v_1,h(v_1)}$ and $x' \in P_{h(v_1),v_1}$, and $P_{x,x'}$ and $P_{x',x}$ are r^*-coverable.*

Proof. As a point x moves along ∂P in counterclockwise direction, $h(x)$ also moves monotonically along ∂P in the same direction; otherwise, $h(x)$ does not satisfy the condition that it is the farthest point or $\tau(x, h(x)) \leq \tau(h(x), x)$.

Consider a point $p \in P_{v_1,h(v_1)}$ such that $P_{p,q}$ and $P_{q,p}$ are r^*-coverable for some point q of ∂P. Then $r^* = \tau(p, h(p)) = \tau(h(p), p)$ since τ is a continuous function. By the monotonicity

of h, $h(v_1) <_p h(p)$. If $\tau(h(v_1), v_1) \neq r^*$, then $r^* = \tau(p, h(p)) < \tau(h(v_1), v_1)$. Thus $h(p) <_p v_1$, implying $h(p) \in P_{h(v_1),v_1}$. Note that $P_{p,h(p)}$ and $P_{h(p),p}$ are r^*-coverable. If $\tau(h(v_1), v_1) = r^*$, $P_{v_1,h(v_1)}$ and $P_{h(v_1),v_1}$ are r^*-coverable. □

Now we partition ∂P into two subchains $P_{v_1,h(v_1)}$ and $P_{h(v_1),v_1}$. We consider x moving along $P_{v_1,h(v_1)}$ and $f(x)$ moving along $P_{h(v_1),v_1}$. For simplicity, we use $<$ for $<_{v_1}$, $\leq$ for $\leq_{v_1}$, $>$ for $>_{v_1}$, and $\geq$ for $\geq_{v_1}$. We need the following technical lemma to compute ROI.

Lemma 9 ([13, 19])**.** *Given any vertex pair (v_i, v_j), we can compute $\tau(v_i, v_j)$ in $O(\log^6 n)$ time, after $O(n \log n)$-time preprocessing.*

If there are two vertices u and v of P such that $P_{u,v}$ and $P_{v,u}$ are r^*-coverable, we can compute r^* by computing the two centers for vertices in $O(n \log n)$ time by using the two-center algorithm for points in convex position [15]. Thus, for the following lemma, we assume $r^* < \max\{\tau(u, v), \tau(v, u))\}$ for every pair of polygon vertices u and v. For simplicity, we use $\tau(i, j)$ for $\tau(v_i, v_j)$ for polygon vertices v_i and v_j.

Lemma 10. *We can compute $O(n/\log^6 n)$ disjoint cells of height and width at most $O(\log^6 n)$ such that every intersection of f^* and g^* lies in one of the cells in $O(n \log n)$ time.*

Proof. Assume that $P_{v_1,h(v_1)}$ has k vertices. Let $m = \lfloor k/\log^6 n \rfloor$, and $p(t) = t \cdot \lfloor k/m \rfloor$ for $t = 1, 2, ..., m-1$. For convenience, let $v_{p(0)} = v_1$ and $v_{p(m)} = h(v_1)$. Let $q(t)$ be any index of a vertex $v_{q(t)}$ such that $f^*(v_{p(t)}) \leq v_{q(t)+1}$ and $g^*(v_{p(t)}) \geq v_{q(t)}$. For every vertex $v_{p(t)}$ from v_1, we find $v_{q(t)}$. We define $F(x) = v_{q(t+1)+1}$ for x with $v_{p(t)} \leq x < v_{p(t+1)}$, and $G(x) = v_{q(t)}$ for x with $v_{p(t)} < x \leq v_{p(t+1)}$.

We compute $q(t)$ as follows. Let r_t be the minimum of $\max\{\tau(p(t), i), \tau(i, p(t))\}$ values over indices $i \in [k+1, n]$. By the assumption that $r^* < \max\{\tau(u, v), \tau(v, u)\}$ for every pair of polygon vertices u and v, we have $r^* < r_t$. Assume that $r_t = \tau(p(t), i)$. Then the largest index $q(t)$ satisfying $\tau(p(t), q(t)) < r_t$ also satisfies $r^* < \tau(p(t), q(t) + 1)$ and $r^* < r_t < \tau(q(t), p(t))$. Thus, $f^*(v_{p(t)}) \leq v_{q(t)+1}$ and $g^*(v_{p(t)}) \geq v_{q(t)}$. The case that $r_t = \tau(i, p(t))$ can be shown by a similar argument.

By Lemma 9, we can find $q(t)$ in $O(\log^7 n)$ time using binary search. Since $m = \lfloor k/\log^6 n \rfloor$, we can compute all $q(t)$'s in $O(n \log n)$ time. Thus, we get m cells $C(t) := [p(t), p(t+1)] \times [q(t), q(t+1)+1]$ for $t \in [0, m-1]$. See Figure 8(b).

We partition each cell $C(t)$ further into subcells as follows. If $q(t+1) - q(t) + 1 \leq \log^6 n$, we simply form a group consisting of at most $\log^6 n$ consecutive indices of polygon vertices, from $q(t)$ to $q(t+1)$. If $q(t+1) - q(t) + 1 > \log^6 n$, we partition the indices of polygon vertices, from $q(t)$ to $q(t+1)$, into disjoint groups consecutively, each consisting of $\log^6 n$ consecutive indices, except for the last group with at most $\log^6 n$ indices. See Figure 8(c) for an illustration for grouping consecutive indices. In this way, we have at most $O(n/\log^6 n)$ groups in total, each consisting of at most $\log^6 n$ consecutive indices in $[q(t), q(t+1)+1]$ for $t \in [0, m-1]$. □

Observe that the cells computed in Lemma 10 form ROI. We say that a vertex pair (v_i, v_j) is in ROI if and only if $i \in [p(t), p(t+1)]$ and $j \in [q(t), q(t+1)+1]$ for some $t \in [0, m-1]$, where $m = \lfloor n/\log^6 n \rfloor$. Also, we say an edge pair $(v_i v_{i+1}, v_j v_{j+1})$ is in ROI if and only if $i, i+1 \in [p(t), p(t+1)]$ and $j, j+1 \in [q(t), q(t+1)+1]$ for some $t \in [0, m-1]$.

4.2 Decision phase

Recall that our parallel decision algorithm finds, for a given r, the intersections of the graphs of $f(x)$ and $g(x)$ in ROI. We use the data structure of the parallel decision algorithm of the

two-center problem for points in convex position [15]. In the parallel decision algorithm, we compute event points in parallel.

Let E be the event points x such that $e(x)$ or $e(f(x))$ or $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ or $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$ changes while x moves along $P_{v_1,h(v_1)}$. Let E' be the subset of E that consists of the event points corresponding to the changes to $e(x)$ or $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$. Let S be the subset of E consisting of the events $x \in E$ such that for any small $\epsilon > 0$, $D_1(x+\epsilon)$ or $D_1(x-\epsilon)$ is of type `T1` or `T2`. To evaluate $f(x)$ for a given r, we compute S for disks of type `T1` or `T2` and all possible disks $D_1(x)$ of type `T3` or `T4`.

To compute S, we first compute the subset $S' = S \cap E'$, by finding $O(n)$ edge pairs $(e(x), e(f(x)))$ in ROI, and then computing the event points in S' in parallel by assigning a processor to each edge. Once we have S', we compute S from S' in parallel by assigning a processor to each event point in S'.

From S, we compute the breakpoints and the corresponding combinatorial structures of $f(x)$ in parallel by assigning a processor to each event point in S. We also do this for $g(x)$. Lastly, for each combinatorial structure, we determine whether there exists $x \in \partial P$ such that $f(x) \geq g(x)$. This process can be done in $O(\log n)$ parallel steps using $O(n)$ processors, after $O(n \log n)$-time preprocessing.

4.2.1 Data structures

We adopt the data structure for the two-center problem for points in convex position by Choi and Ahn [15]. To construct the data structure, they store a set of intersections of disks for all $r > 0$ that are frequently used for queries. Then, they find a range of radii $(r_1, r_2]$ containing the optimal radius r' for the two-center problem for points in convex position. To do this they use binary search and the sequential decision algorithm for points in convex position. In our case, we compute a range of radii $(r_1, r_2]$ containing the optimal radius r^* using binary search and the sequential decision algorithm in Section 3 running in $O(n)$ time. For $r \in (r_1, r_2]$, we construct a data structure that supports the following.

Lemma 11 ([15])**.** *After $O(n \log n)$-time preprocessing, we can construct a data structure in $O(\log n)$ parallel steps using $O(n)$ processors that supports the following queries with $r \in (r_1, r_2]$: (1) Given a vertex v_i in $P_{v_1,h(v_1)}$, compute $I(v_i, h(v_1))$ represented in a binary search tree of height $O(\log n)$ in $O(\log n)$ time. (2) Given a pair (v_i, v_j) of vertices in ROI, determine if $I(v_i, v_j) = \emptyset$ in $O(\log^3 \log n)$ time.*

4.2.2 Computing edge pairs

Using the data structure in Lemma 11, we can compute all edges pairs $(e(x), e(f(x)))$ in ROI efficiently, as in the following lemma.

Lemma 12. *Given $r \in (r_1, r_2]$, we can compute all edge pairs $(e(x), e(f(x)))$ in ROI in $O(\log n)$ parallel time using $O(n)$ processors, after $O(n \log n)$-time preprocessing.*

Proof. By Lemma 11, we can construct a data structure supporting the following query: given a pair of vertices (v_i, v_j) in ROI, determine if $I_r(v_i, v_j) = \emptyset$ in $O(\log^3 \log n)$ time. For each vertex v_i, we find a vertex v_j such that (v_i, v_j) is in ROI, $I_r(v_i, v_j) \neq \emptyset$, and $I_r(v_i, v_{j+1}) = \emptyset$ in $O(\log^4 \log n)$ time using the data structure and performing binary search on $\log^6 n$ vertices. It takes $O(\log n)$ parallel time using $O(n)$ processors to construct the data structure, after $O(n \log n)$-time preprocessing. Thus, we find all edge pairs in ROI in $O(\log n)$ parallel time using $O(n)$ processors, after $O(n \log n)$-time preprocessing. □

4.2.3 Computing the combinatorial structure

After computing the edge pairs using Lemma 12, we compute the breakpoints and the corresponding combinatorial structures of $f(x)$. To do this, we compute event points and find breakpoints from the event points for each edge pair. For $D_1(x)$ of type T3 or T4, its determinators never change for an edge pair $(e(x), e(f(x)))$ by Lemma 2. Thus, for each edge pair we find candidates of the determinators of $D_1(x)$ of type T3 or T4 in $O(\log n)$ time.

For $D_1(x)$ of type T1 or T2, we find the event points of E' and the event points of S. Consider an edge pair $(u'u, vv')$ in ROI such that $f(x) \in vv'$ for some $x \in u'u$. The edge pair $(u'u, vv')$ may have $O(n)$ event points at which $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ or $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$ changes, while the total number of event points is $O(n)$. We find the event points x of $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ in $u'u$ using $I(u, h(v_1))$ while $D_1(x)$ is of type T1 or T2. Since $I(u, h(v_1))$ is represented in a binary search tree, this can be done in $O(\log n)$ time. Thus, we can find S' for all edge pairs in $O(\log n)$ parallel steps using $O(n)$ processors. For two event points of S' consecutive along $P_{v_1,h(v_1)}$, we compute the corresponding event points of $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$ in $O(\log n)$ time. For a segment ab such that a and b are event points of S consecutive along $P_{v_1,h(v_1)}$, we compute $f(x)$.

Lemma 13. *Given $r \in (r_1, r_2]$, we can compute $f(x)$ for all $x \in \partial P$ such that $(e(x), e(f(x)))$ is an edge pair in ROI, represented in a binary search tree of height $O(\log n)$ consisting of $O(n)$ nodes, in $O(\log n)$ parallel steps using $O(n)$ processors, after $O(n \log n)$-time preprocessing.*

Proof. Let $(u'u, vv')$ be an edge pair in ROI such that $f(x) \in vv'$ for some $x \in u'u$. Imagine that x moves along $u'u$ from u' to u. We compute $f(x)$ for each type of $D_1(x)$. Consider the case that $D_1(x)$ is of type T3 or T4. Let $x' \in u'u$ be the first breakpoint of $f(x)$ from u' such that $D_1(x)$ becomes a disk of type T3 or T4. Then the determinators of $D_1(x)$ remain the same for any $x \in x'u$ by Lemma 2. Thus, $P_{u,f(x)}$ has the same determinators and $I(u, f(x))$ must be a point. This implies that the intersection of $I(u, v)$ and the disk of radius r centered at $f(x)$ is just a point. We can find $I(u, v)$ and $f(x)$ in $O(\log n)$ time by Lemma 11 and binary search. If the intersection point is a vertex w of $I(u, v)$, then $P_{u,f(x)}$ is covered by $D_1(x)$ of type T4. Moreover, the two vertices of $P_{u,f(x)}$ defining w are the two determinators of $D_1(x)$ other than $f(x)$. See Figure 9(a) for an illustration. If the intersection point is in an arc of $I(u, v)$, then $P_{u,f(x)}$ is covered by $D_1(x)$ of type T3, and the other determinator is the vertex of $P_{u,f(x)}$ defining the arc.

For disks $D_1(x)$ of type T1 or T2, we find the event points in S as follows. To find the event points of $e(f(x))$, we first find the maximal segment ss' contained in $u'u$ such that $e(f(x))$ is vv' for any $x \in ss'$. Let $D(x)$ be the disk of radius r centered at x. Note that s in $e(x)$ is the closest point from u' such that $D(s) \cap I(u, v) \neq \emptyset$, and s' in $e(x)$ is the farthest point from u' such that $D(s') \cap I(u, v') \neq \emptyset$. We can find s and s' in $O(\log n)$ time by binary search. See Figure 9(b) for an illustration.

For $x \in ss'$, we find the event points of $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ using $I(u, h(v_1))$. The vertices of P that are the center of the arcs of $\alpha(u, x_1)$ lying in between $\partial D(s) \cap \partial I(u, h(v_1))$ and $\partial D(s') \cap \partial I(u, h(v_1))$ are $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ while x moves along ss'. This is due to the duality between $I(x, h(v_1))$ and $\alpha(x, h(v_1))$. See Figure 9(c) for an illustration.

Similarly, we find the maximal segment tt' such that $e(x)$ is $u'u$ for all $f(x) \in tt'$, and compute $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$ for $f(x) \in tt'$ in $O(\log n)$ time. Thus, we can find the event points contained in all edge pairs in $O(\log n)$ time using $O(n)$ processors.

Let z and z' be two event points of $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ consecutive along $P_{v_1,h(v_1)}$ such that zz' is contained in $e(x)$. We can find $\mathsf{cw}(f(z))$ of $\alpha(h(v_1), f(z))$ and $\mathsf{cw}(f(z'))$ of $\alpha(h(v_1), f(z'))$ in $O(\log n)$ time by binary search over the event points of $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$. The event points of $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$ lying in between $f(z)$ and $f(z')$ are represented as a binary

search tree of height $O(\log n)$. We repeat this process for every segment in $e(x)$ connecting two event points of $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ consecutive along $P_{v_1,h(v_1)}$. Then we have all event points of $\mathsf{cw}(f(x))$ of $\alpha(h(v_1), f(x))$ lying in between two event points of of $\mathsf{ccw}(x)$ of $\alpha(x, h(v_1))$ consecutive along $P_{v_1,h(v_1)}$. By Lemma 4 and Corollary 1, there are $O(n)$ event points in total.

Let ab be a segment such that a and b are event points in S consecutive along $P_{v_1,h(v_1)}$. By comparing two $f(x)$ functions, one for the current type of $D_1(x)$ and one for other types of $D_1(x)$, we can compute in $O(1)$ time the next breakpoint, the new determinators, and the type of $D_1(x)$ for x right after the breakpoint. Therefore, we can compute $f(x)$ for all $x \in \partial P$ such that $(e(x), e(f(x)))$ is an edge pair in ROI, represented in a binary search tree of height $O(\log n)$ consisting of $O(n)$ nodes, in $O(\log n)$ parallel steps using $O(n)$ processors, after $O(n \log n)$-time preprocessing. □

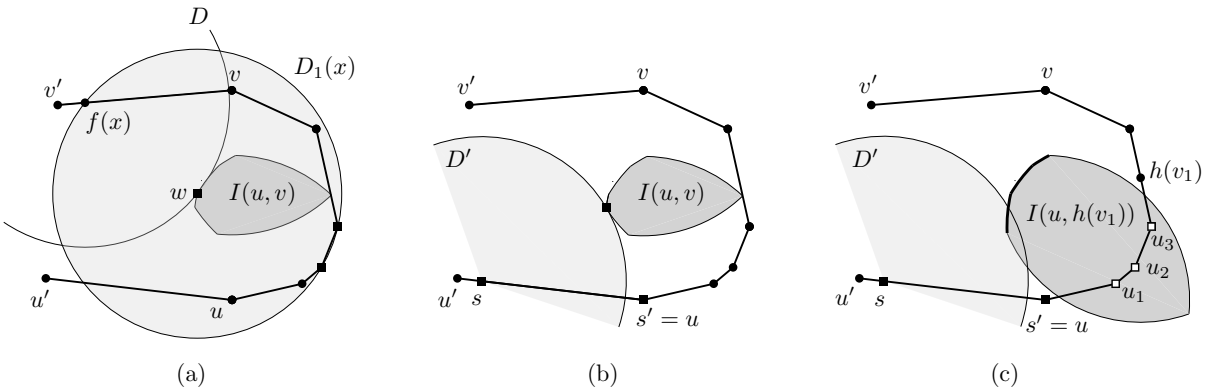

Figure 9: Computing the candidates of $D_1(x)$. (a) $D_1(x)$ is of type `T4`. The intersection of $I(u, v)$ and the disk D of radius r centered at $f(x)$ is just a vertex w of $I(u, v)$. (b) Finding a maximal segment $[s, s']$ for $D_1(x)$ of type `T1`. D' is the disk of radius r centered at s. (c) Finding $\mathsf{ccw}(x)$'s of $\alpha(x, h(v_1))$ while x moves along $[s, s']$. The bold arcs of $I(u, h(v_1))$ are the arcs defined by $\mathsf{ccw}(x)$'s. $u_1 = \mathsf{ccw}(s'), u_3 = \mathsf{ccw}(s)$.

Now, we have $f(x)$ and $g(x)$ within ROI, each represented in a binary search tree of height $O(\log n)$ and size $O(n)$. For two consecutive breakpoints t and t' of $f(x)$, we find the corresponding combinatorial structures of $g(t)$ and $g(t')$. Then we determine whether there exists $x \in tt'$ such that $f(x) \geq g(x)$ for all combinatorial structures of $g(x)$. Since $f(x)$ and $g(x)$ have $O(n)$ breakpoints by Lemma 5, we can determine whether there are two disks of radius r covering P in $O(\log n)$ parallel steps using $O(n)$ processors, after $O(n \log n)$-time preprocessing. Therefore, using Lemma 13, we get the following theorem.

Theorem 2. *Given a real value r, we can determine whether $r \geq r^*$ in $O(\log n)$ parallel steps using $O(n)$ processors, after $O(n \log n)$-time preprocessing.*

By applying Cole's parametric search [16] with our sequential decision algorithm and parallel decision algorithm, we get the following theorem.

Theorem 3. *Given a convex polygon with n vertices in the plane, we can find in $O(n \log n)$ time two congruent disks of minimum radius whose union covers the polygon.*

Proof. We use Cole's parametric search technique [16] to compute the optimal radius r^*. For a sequential decision algorithm of running time T_S and a parallel decision algorithm of parallel running time T_P using N processors, we can apply Cole's parametric search and compute r^* in $O(NT_P + T_S(T_P + \log N))$ time. To apply Cole's parametric search, the parallel decision

algorithm must satisfy the bounded fan-in/fan-out requirement. Our parallel decision algorithm satisfies this requirement because it consists of independent binary searches. In our case, $T_S = O(n)$, $T_P = O(\log n)$, and $N = O(n)$. Therefore, by applying Cole's technique, r^* can be computed in $O(n \log n)$ time. □

5 Conclusions

We present an $O(n \log n)$-time algorithm for the two-center problem for a convex polygon, where n is the number of vertices of the polygon. However, it is unknown whether the time complexity is optimal. Thus, a natural question is whether there is any lower bound other than the trivial $\Omega(n)$. We would like to mention that our sequential decision algorithm can be used for covering the boundary of a convex polygon (and any curve) using the minimum number of unit disks under the condition that a unit disk covers at most one connected component of the boundary.

References

[1] P. K. Agarwal, R. Ben Avraham, and M. Sharir. The 2-center problem in three dimensions. *Computational Geometry: Theory and Applications*, 46(6):734–746, 2013.

[2] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[3] P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72(1):83–98, 2015.

[4] P. K. Agarwal and M. Sharir. Planar geometric location problems. *Algorithmica*, 11(2):185–195, 1994.

[5] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.

[6] 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 & Computational Geometry*, 4(6):591–604, 1989.

[7] H.-K. Ahn, L. Barba, P. Bose, J.-L. D. Carufel, M. Korman, and E. Oh. A linear-time algorithm for the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 56(4):836–859, 2016.

[8] H.-K. Ahn, H.-S. Kim, S.-S. Kim, and W. Son. Computing k centers over streaming data for small k. *International Journal of Computational Geometry and Applications*, 24(2):107–123, 2014.

[9] H.-K. Ahn, S.-S. Kim, C. Knauer, L. Schlipf, C.-S. Shin, and A. Vigneron. Covering and piercing disks with two centers. *Computational Geometry: Theory and Applications*, 46(3):253–262, 2013.

[10] S. W. Bae. L_1 geodesic farthest neighbors in a simple polygon and related problems. *Discrete & Computational Geometry*, 62(4):743–774, 2019.

[11] M. Basappa, R. K. Jallu, and G. K. Das. Constrained k-center problem on a convex polygon. *International Journal of Foundations of Computer Science*, 31(02):275–291, 2020.

[12] T. Chan and V. Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Computational Geometry: Theory and Applications*, 47(2):240–247, 2014.

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

[14] K. Cho and E. Oh. Optimal algorithm for the planar two-center problem. *arXiv:2007.08784*, 2020.

[15] J. Choi and H.-K. Ahn. Efficient planar two-center algorithms. *Computational Geometry: Theory and Applications*, 97:101768, 2021.

[16] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, 1987.

[17] G. K. Das, S. Roy, S. Das, and S. C. Nandy. Variations of base-station placement problem on the boundary of a convex region. *International Journal of Foundations of Computer Science*, 19(02):405–427, 2008.

[18] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559, 1983.

[19] D. Eppstein. Faster construction of planar two-centers. In *Proceedings of the 8th annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997)*, pages 131–138, 1997.

[20] T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 434–444, 1988.

[21] K. Fischer and B. Gärtner. The smallest enclosing ball of balls: combinatorial structure and algorithms. *International Journal of Computational Geometry & Applications*, 4(5):341–378, 2004.

[22] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[23] J. Hershberger and S. Suri. Finding tailored partitions. *Journal of Algorithms*, 12(3):431–463, 1991.

[24] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. *Computational Geometry: Theory and Applications*, 39(3):191–208, 2008.

[25] R. Hwang, R. C. T. Lee, and R. Chang. The slab dividing approach to solve the euclideanp-center problem. *Algorithmica*, 9(1):1–22, 1993.

[26] S. K. Kim and C.-S. Shin. Efficient algorithms for two-center problems for a convex polygon. In *Proceedings of the 6th Annual International Conference on Computing and Combinatorics (COCOON 2000)*, pages 299–309, 2000.

[27] S.-S. Kim and H.-K. Ahn. An improved data stream algorithm for clustering. *Computational Geometry: Theory and Applications*, 48(9):635–645, 2015.

[28] M. Löffler and M. Van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry: Theory and Applications*, 43(4):419–433, 2010.

[29] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM journal on computing*, 12(4):759–776, 1983.

[30] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, 1984.

[31] N. Megiddo. On the ball spanned by balls. *Discrete & Computational Geometry*, 4(1):605–610, 1989.

[32] E. Oh, J.-L. De Carufel, and H.-K. Ahn. The geodesic 2-center problem in a simple polygon. *Computational Geometry: Theory and Applications*, 74:21–37, 2018.

[33] S. Roy, D. Bardhan, and S. Das. Base station placement on boundary of a convex polygon. *Journal of Parallel and Distributed Computing*, 68(2):265–273, 2008.

[34] S. Sadhu, S. Roy, S. Nandi, A. Maheshwari, and S. C. Nandy. Two-center of the convex hull of a point set: Dynamic model, and restricted streaming model. *Fundamenta Informaticae*, 164(1):119–138, 2019.

[35] M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete & Computational Geometry*, 18(2):125–134, 1997.

[36] C.-S. Shin, J.-H. Kim, S. K. Kim, and K.-Y. Chwa. Two-center problems for a convex polygon. In *Proceedings of the 6th annual European Symposium on Algorithms (ESA 1998)*, pages 199–210, 1998.

[37] H. Wang. On the planar two-center problem and circular hulls. In *Proceeding of the 36th International Symposium on Computational Geometry (SoCG 2020)*, pages 68:1–68:14, 2020.

[38] H. Wang. On the planar two-center problem and circular hulls. *arXiv preprint arXiv:2002.07945*, 2020.

[39] H. Zarrabi-Zadeh. Core-preserving algorithms. In *Proceedings of the 20th annual Canadian Conference on Computational Geometry (CCCG 2008)*, pages 159–162, 2008.