




# Covering Convex Polygons by Two Congruent Disks

Jongmin Choi<sup>1(✉)</sup>, Dahye Jeong<sup>1(✉)</sup>, and Hee-Kap Ahn<sup>2(✉)</sup> 

<sup>1</sup> Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea

{icothos,dahyejeong}@postech.ac.kr

<sup>2</sup> Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Korea

heekap@postech.ac.kr

**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.

**Keywords:** Two-center problem · Covering · Convex polygon

## 1 Introduction

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 is to find  $k$  homothets<sup>1</sup> of  $Q$  with the same homothety ratio such that their union contains  $R$  and the homothety ratio is minimized. The homothets in the covering are allowed to overlap, as long as their union contains the region. This is a fundamental optimization problem [2, 4, 19] arising in analyzing and recognizing shapes, and it has real-world applications, 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 last decades, there have been a lot of works, including exact algorithms for  $k = 2$  [3, 11, 13, 14, 33, 35], exact

---

<sup>1</sup> 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.

---

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)).

© Springer Nature Switzerland AG 2021

P. Flocchini and L. Moura (Eds.): IWOCA 2021, LNCS 12757, pp. 165–178, 2021.

[https://doi.org/10.1007/978-3-030-79987-8\\_12](https://doi.org/10.1007/978-3-030-79987-8_12)

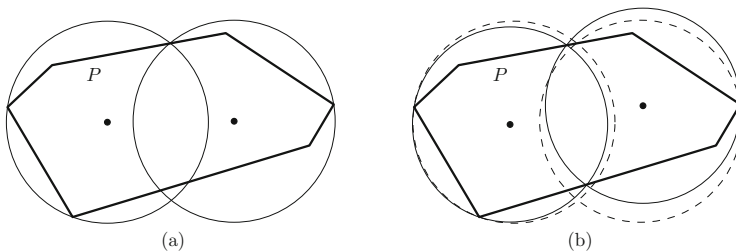
and approximation algorithms for large  $k$  [2, 19, 21, 24], algorithms in higher dimensional spaces [1, 2, 29], and approximation algorithms for streaming points [5, 6, 12, 22, 25, 37]. There are also some works on the  $k$ -center problem for small  $k$  when the region to cover is a set of disks in the plane, for  $k = 1$  [20, 27, 28] and  $k = 2$  [8].

In the context of the facility location, there have also been some works on the *geodesic  $k$ -center* problem for simple polygons [7, 30] and polygonal domains [9], in which we find  $k$  points (centers) 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 Fig. 1 for an illustration.

*Previous Works.* For a convex polygon with  $n$  vertices, Shin et al. [34] 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 centers, including the centers restricted to lie on the polygon boundary [31] and on a given polygon edge(s) [17, 31]. 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. [10] gave a  $(2 + \epsilon)$ -approximation algorithm for  $k \geq 7$ , where the centers are restricted to lie on the polygon boundary. There is a 2-approximation algorithm for the two-center problem for a convex polygon that supports insertions and deletions of points in  $O(\log n)$  time per operation [32].



**Fig. 1.** (a) Two congruent disks whose union covers a convex polygon  $P$ . (b)  $P$  can be covered by two congruent disks of smaller radius.

*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].

*Sketch of Our Algorithm.* Our algorithm is twofold. First we solve the sequential decision problem in  $O(n)$  time. That is, given a real value  $r$ , decide whether  $r \geq r^*$ , where  $r^*$  is the optimal radius value. 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, the two centers for  $P$ , 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 that define 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  $x'$  in  $P$  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  $x'$  in  $P$  such that the two longest subchains from  $x'$  covers 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 [14] 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  $x'$  in  $P$  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  $P$ .

Due to lack of space, some of the proofs and details are omitted. A full version of this paper is available in [15].

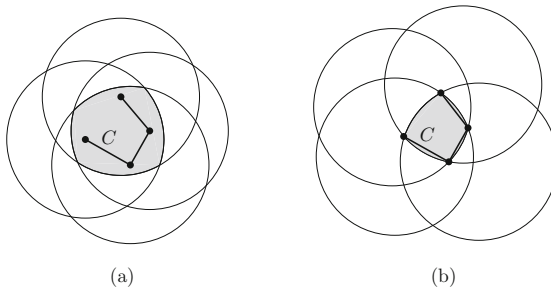
## 2 Preliminaries

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  $\partial A$  to denote the boundary of  $A$ . We simply say  $x$  moves along  $\partial A$  when  $x$  moves in the counterclockwise direction along  $\partial A$ . Otherwise, we explicitly mention the direction.

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

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

For a subchain  $C$  of  $\partial P$ , we denote by  $I_r(C)$  the intersection of the disks of radius  $r$ , each centered at a point in  $C$ . See Fig. 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 Fig. 2(b). Let  $S$  be the set of vertices of a subchain  $C$  of  $\partial P$ . If a disk covers  $C$ , it also covers  $S$ . If a disk covers  $S$ , it covers  $C$  since it covers every line segment induced by pairs of the points in  $S$ , 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.



**Fig. 2.**  $C$  is a subchain of  $\partial P$  and  $S$  is the vertex set of  $C$ . (a)  $I_r(S) = I_r(C)$  (b)  $\alpha_r(S) = \alpha_r(C)$

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.

For a vertex  $v$  of  $\alpha_r(C)$ , we denote by  $\text{ccw}(v)$  its counterclockwise neighbor on  $\partial\alpha_r(C)$ , and by  $\text{cw}(v)$  its clockwise neighbor on  $\partial\alpha_r(C)$ . We denote by  $\gamma(v)$  the arc of  $\alpha_r(C)$  connecting  $v$  and  $\text{ccw}(v)$  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  $\partial P$  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  $\partial P$  from  $x$  in the counterclockwise direction along  $\partial P$  such that  $P_{x, f_r(x)}$  is  $r$ -coverable. We denote by  $D_1^r(x)$  the disk of radius  $r$  covering  $P_{x, f_r(x)}$ . Similarly, let  $g_r(x)$  be the farthest point on  $\partial P$  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  $r$  covering  $P_{g_r(x), x}$ . 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.

Since we can determine in  $O(n)$  time whether  $P$  is  $r$ -coverable [29], 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  $\partial P$  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  $\partial P$  in the counterclockwise direction, both  $f(x)$  and  $g(x)$  move monotonically along  $\partial P$  in the counterclockwise direction.

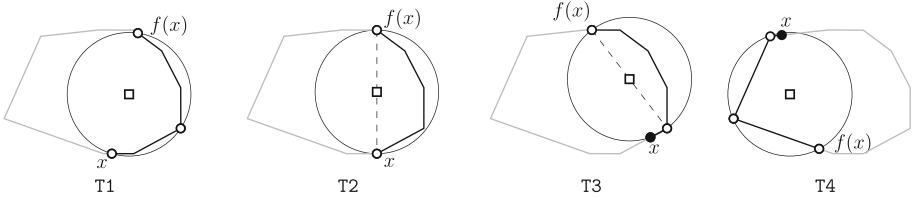
### 3 Sequential Decision Algorithm

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  $\partial P$ , 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  $\partial P$ . 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  $\partial P$ , 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 it 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 defined 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 Fig. 3 for an illustration of the four types.



**Fig. 3.** Four types of  $D_1(x)$  and its determinators (small circles).

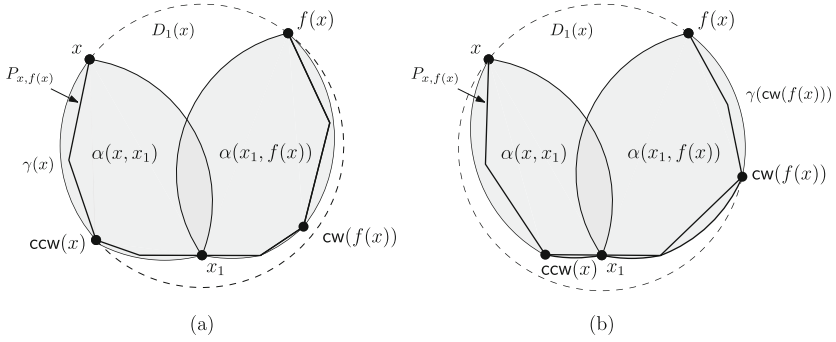
We denote by  $e(a)$  the edge of  $P$  containing a point  $a \in \partial P$ . If  $a$  is a vertex of  $P$ ,  $e(a)$  denotes the edge of  $P$  incident to  $a$  lying in the counterclockwise direction from  $a$ . For a point  $x$  moving along  $\partial P$ , the *combinatorial structure* of  $f(x)$  is determined by  $e(x)$ ,  $e(f(x))$ , and the determinators of  $D_1(x)$ . We call each point  $x$  in  $\partial P$  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  $D_1(x)$ .

Consider  $x$  moving along  $\partial P$  starting from  $x_0$  on  $\partial P$  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  $i = 0, \dots, 3$  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 handle 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  $\partial P$ .*

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 a breakpoint of  $f(x)$  for  $x$  moving along  $P_{0,1}$  and a structural change to  $\alpha(x, f(x))$ . This is 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 Fig. 4 for an illustration.

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



**Fig. 4.** Two cases of  $D_1(x)$  of type T1. Two arcs (dashed) of  $D_1(x)$  connecting  $\alpha(x, x_1)$  and  $\alpha(x_1, f(x))$ . (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  $\delta(\text{cw}(f(x)))$ .

**Lemma 2.** *The following characterizations hold for each type of  $D_1(x)$ .*

- For  $D_1(x)$  of type T1, it is  $\delta(x)$  of  $\alpha(x, x_1)$  or  $\delta(\text{cw}(f(x)))$  of  $\alpha(x_1, f(x))$ .
- For  $D_1(x)$  of type T2, the Euclidean distance between  $x$  and  $f(x)$  is  $2r$ .
- For  $D_1(x)$  of type T3 or T4 containing  $x$  on its boundary, it is  $\delta(x)$  of  $\alpha(x, x_1)$  or  $\delta(\text{cw}(f(x)))$  of  $\alpha(x_1, f(x))$ . Moreover, for any point  $y$  in the interior of  $P_{x,v}$ ,  $D_1(y)$  has the same type as  $D_1(x)$ , where  $v$  is the determinant of  $D_1(x)$  closest to  $x$  in counterclockwise order.

If there is a change to  $e(x)$ ,  $e(f(x))$ ,  $\text{ccw}(x)$  of  $\alpha(x, x_1)$  or  $\text{cw}(f(x))$  of  $\alpha(x_1, f(x))$ , the combinatorial structure of  $f(x)$  changes. Therefore, we compute the changes to  $e(x)$  and  $\text{ccw}(x)$  of  $\alpha(x, x_1)$  for a point  $x$  moving along  $P_{0,1}$ , and compute the changes to  $e(y)$  and  $\text{cw}(y)$  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

Wang [36] proposed a semi-dynamic (insertion-only) data structure for maintaining the circular hull for points in the plane that are inserted in increasing order of their  $x$ -coordinates. It is also mentioned that the algorithm can be modified to work for points that are inserted in the sorted order around a point. Since the vertices of  $P$  are already sorted around any point in the interior of  $P$ , we can use the algorithm for our purpose.

**Lemma 3 (Theorem 5 in [36]).** *We can maintain the circular hull of a set  $Q$  of points such that when a new point to the right of all points of  $Q$  is inserted, we can decide in  $O(1)$  amortized time whether  $\alpha(Q)$  is nonempty, and update  $\alpha(Q)$ .*

We can modify the algorithm to work not only for point insertions, but also for edge insertions. Let  $v_1, \dots, v_i$  be the vertices of  $P$  inserted so far in order from  $v_1$ . When  $v_{i+1}$  is inserted, we compute the points  $z$  on edge  $v_i v_{i+1}$  at which a structural change to  $\alpha(v_1, z)$  occurs.

**Lemma 4.** *For a point  $x$  moving along  $P_{0,1}$ ,  $e(x)$  and  $\text{ccw}(x)$  of  $\alpha(x, x_1)$  change  $O(|P_{0,1}|)$  times. We can compute the event points  $x$  at which  $\text{ccw}(x)$  of  $\alpha(x, x_1)$  changes in  $O(|P_{0,1}|)$  time.*

From Lemma 4, we obtain the following Corollary.

**Corollary 1.** *For a point  $y$  moving along  $P_{1,2}$ ,  $e(y)$  and  $\text{cw}(y)$  of  $\alpha(x_1, y)$  change  $O(|P_{1,2}|)$  times. We can compute the event points  $y$  at which  $\text{cw}(y)$  of  $\alpha(x_1, y)$  changes in  $O(|P_{1,2}|)$  time.*

The event points subdivide  $P_{0,1}$  and  $P_{1,2}$  into  $O(|P_{0,1}|)$  and  $O(|P_{1,2}|)$  pieces, respectively. Since the vertices of  $P_{0,1}$  and  $P_{1,2}$  are also event points (defined by the changes to  $e(x)$  and  $e(y)$ ), each piece is a segment contained in an edge. Moreover, any point  $x$  in a segment of  $P_{0,1}$  has the same  $\text{ccw}(x)$  of  $\alpha(x, x_1)$ , and any point  $y$  in a segment of  $P_{1,2}$  has the same  $\text{cw}(y)$  of  $\alpha(x_1, y)$ .

Let  $T$  be a maximal segment contained in an edge of  $P_{0,1}$  such that  $e(x)$ ,  $e(f(x))$ ,  $\text{ccw}(x)$  of  $\alpha(x, x_1)$ , and  $\text{cw}(f(x))$  of  $\alpha(x_1, f(x))$  remain the same for any  $x \in T$ . We count the breakpoints of  $f(x)$  in the interior of  $T$ . There are  $O(n)$  such segments by Lemmas 1, 4 and Corollary 1. We count the breakpoints of  $f(x)$  by computing point  $x$  where the type of  $D_1(x)$  or the determinators of  $D_1(x)$  changes. We show that there are at most  $O(1)$  breakpoints in the interior of each maximal segment, and therefore there are  $O(n)$  breakpoints in total. In order to compute  $f(x)$ , we first compute  $f(x_0) = x_1$ . Then starting from  $x = x_0$  and  $f(x) = x_1$ , we compute  $f(x)$  as  $x$  moves along  $\partial P$  by maintaining the two maximal segments such that  $e(x)$  and  $\text{ccw}(x)$  of  $\alpha(x, x_1)$  remain the same and  $e(f(x))$  and  $\text{cw}(f(x))$  of  $\alpha(x_1, f(x))$  remain the same. By repeating this process over maximal segments, we get the following lemma. The details of the process can be found in [15].

**Lemma 5.** *For a fixed  $r \geq r^*$ , there are  $O(n)$  breakpoints of  $f(x)$  and  $g(x)$ , and they 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 5, 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 longest subchains of  $\partial P$  from  $x$  covered by disks of radius  $r$ , and 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 [14] 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  $r$ . 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, two congruent disks must cover the edges of  $P$  as well as the vertices of  $P$ , and thus we modify the preprocessing phase.

In the preprocessing phase, their algorithm partitions the vertices of  $P$  into two subsets  $S_1 = \{v_1, \dots, v_k\}$  and  $S_2 = \{v_{k+1}, \dots, v_n\}$ , each consisting of consecutive vertices along  $\partial P$  such that there are  $v_i \in S_1$  and  $v_j \in S_2$  satisfying  $\{v_i, v_{i+1}, \dots, v_{j-1}\} \subset D_1$  and  $\{v_j, v_{j+1}, \dots, v_{i-1}\} \subset D_2$  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.

In the preprocessing phase, our algorithm partitions  $\partial P$  into two subchains. Then, we partition  $\partial P$  into  $O(n/\log^6 n)$  subchains, each consisting of  $O(\log^6 n)$  consecutive vertices, and compute  $O(n/\log^6 n)$  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.

In the decision phase, their algorithm constructs a data structure in  $O(\log n)$  parallel time with  $O(n)$  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.

In our case, our algorithm 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  $I(v_i, v_j) = \emptyset$ .

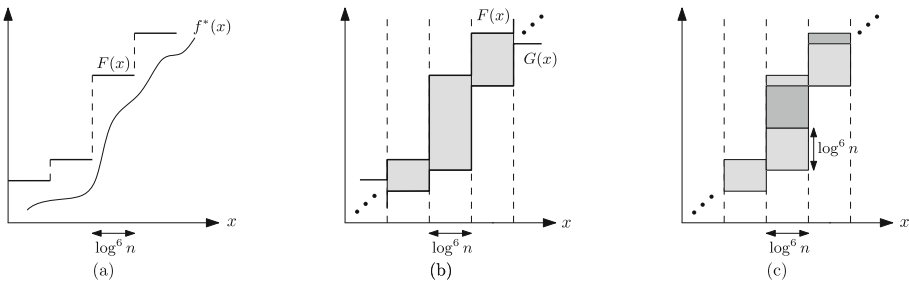
Using the data structure, our algorithm gets rough bounds of  $f(x)$  and  $g(x)$ . Then it computes  $f(x)$  and  $g(x)$  exactly. In doing so, it computes all break-points of  $f(x)$  and  $g(x)$ , and their corresponding combinatorial structures, and determines whether there exists  $x \in \partial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

### 4.1 Preprocessing Phase

We use  $f^*(x)$  and  $g^*(x)$  to denote  $f_{r^*}(x)$  and  $g_{r^*}(x)$ , respectively. Our algorithm partitions  $\partial P$  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  $\partial P$ , each consisting of  $O(\log^6 n)$  consecutive vertices.

More precisely, for any two points  $x, y \in \partial P$ , let  $\tau(x, y)$  be the smallest value such that  $P_{x,y}$  is  $\tau(x, y)$ -coverable. For a point  $p \in \partial P$ , let  $h(p)$  be the farthest point from  $p$  in counterclockwise direction along  $\partial P$  that satisfies  $\tau(p, h(p)) \leq \tau(h(p), p)$ . Then for any vertex  $v$  of  $P$ ,  $P_{v,h(v)}$  and  $P_{h(v),v}$  form a partition of  $\partial P$  such that there are  $x \in P_{v,h(v)}$  and  $x' \in P_{h(v),v}$ , and  $P_{x,x'}$  and  $P_{x',x}$  are  $r^*$ -coverable. The details on the partition of  $\partial P$  into two subchains can be found in [15].

We consider  $x$  moving along  $P_{v_1,h(v_1)}$  and  $f(x)$  moving along  $P_{h(v_1),v_1}$ . Also, from now on, we use  $<$  instead of  $<_{v_1}$ . The same goes for  $\leq_{v_1}, >_{v_1}, \geq_{v_1}$ . To compute rough bounds of  $f(x)$  and  $g(x)$ , 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  $\partial P$ , it evaluates step functions  $F(v)$  and  $G(v)$  on  $r^*$  such that  $f^*(v) \leq F(v)$  and  $g^*(v) \geq G(v)$ . See Fig. 5(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 Fig. 5(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 Fig. 5(c).



**Fig. 5.** (a) Step function  $F(x)$  satisfying  $f^*(x) \leq F(x)$ , with intervals, each consisting of  $\log^6 n$  consecutive vertices. (b) Sequence of rectangular cells bounded in between  $F(x)$  and  $G(x)$ . (c) Disjoint rectangular cells of width and height at most  $\log^6 n$ .

We say that a vertex pair  $(v_i, v_j)$  is in ROI if and only if  $G(v_i) \leq v_j \leq F(v_i)$ . Also, we say an edge pair  $(v_i v_{i+1}, v_j v_{j+1})$  is in ROI if and only if vertex pairs  $(v_i, v_j)$ ,  $(v_i, v_{j+1})$ ,  $(v_{i+1}, v_j)$  and  $(v_{i+1}, v_{j+1})$  are all in ROI. The details on computing ROI can be found in [15].

## 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 [14]. To evaluate  $f(x)$  for a given  $r$ , we first find  $O(n)$  edge pairs  $(e(x), e(f(x)))$  in ROI. Then we assign a processor to each edge pair to compute the event points. Then we assign a processor to each event point to compute the breakpoints and the corresponding combinatorial structures of  $f(x)$ . 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.

**Data Structures.** We adopt the data structure for the two-center problem for points in convex position by Choi and Ahn [14]. To construct the data structure, they store the frequently used intersections of disks for all  $r > 0$ . 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 Sect. 3 running in  $O(n)$  time. For  $r \in (r_1, r_2]$ , we construct a data structure that supports the following.

**Lemma 6** [14]. *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) For any vertex  $v_i$  in  $P_{v_1, h(v_1)}$ , compute  $I(v_i, h(v_1))$  represented in a binary search tree with height  $O(\log n)$  in  $O(\log n)$  time. (2) For any pair  $(v_i, v_j)$  of vertices in ROI, determine if  $I(v_i, v_j) = \emptyset$  in  $O(\log^3 \log n)$  time.*

**Computing Edge Pairs.** Using the data structure in Lemma 6, we get the following lemma.

**Lemma 7.** *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.*

**Computing the Combinatorial Structure.** After computing the edge pairs using Lemma 7, 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  $\text{ccw}(x)$  of  $\alpha(x, h(v_1))$ , and the event points of  $\text{cw}(f(x))$  of  $\alpha(h(v_1), f(x))$ . 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  $\text{ccw}(x)$  of  $\alpha(x, h(v_1))$  or  $\text{cw}(f(x))$  of  $\alpha(h(v_1), f(x))$  changes, while the total number of event points is  $O(n)$ . We find the event points of  $\text{ccw}(x)$  of  $\alpha(x, h(v_1))$  represented in a binary search tree using  $I(u, h(v_1))$  in  $O(\log n)$  time. Thus, we can find the event points of  $\text{ccw}(x)$  of  $\alpha(x, h(v_1))$  for all edge pairs in  $O(\log n)$  parallel steps using  $O(n)$  processors. For two consecutive event points of  $\text{ccw}(x)$  of  $\alpha(x, h(v_1))$ , we compute the corresponding event points of  $\text{cw}(f(x))$  of  $\alpha(h(v_1), f(x))$  in  $O(\log n)$  time. For a segment  $T$  such that  $e(x)$ ,  $\text{ccw}(x)$  of  $\alpha(x, h(v_1))$ ,  $e(f(x))$ , and  $\text{cw}(f(x))$  of  $\alpha(h(v_1), f(x))$  remain the same for any  $x \in T$ , we compute  $f(x)$ .

**Lemma 8.** *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 as 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.*

Now, we have  $f(x)$  and  $g(x)$  within ROI, each represented as 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 two disks of radius  $r$  cover  $P$  in  $O(\log n)$  parallel steps using  $O(n)$  processors, after  $O(n \log n)$ -time preprocessing. Therefore, using Lemma 8, 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.*

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 to 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 a bounded fan-in/bounded fan-out requirement. Our parallel decision algorithm satisfies such requirement. 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.

**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.*

## References

1. Agarwal, P.K., Ben Avraham, R., Sharir, M.: The 2-center problem in three dimensions. *Computat. Geom. Theor. Appl.* **46**(6), 734–746 (2013)
2. Agarwal, P.K., Procopiuc, C.M.: Exact and approximation algorithms for clustering. *Algorithmica* **33**(2), 201–226 (2002). <https://doi.org/10.1007/s00453-001-0110-y>
3. Agarwal, P.K., Sharir, M.: Planar geometric location problems. *Algorithmica* **11**(2), 185–195 (1994). <https://doi.org/10.1007/BF01182774>
4. Agarwal, P.K., Sharir, M.: Efficient algorithms for geometric optimization. *ACM Comput. Surv.* **30**(4), 412–458 (1998)
5. Agarwal, P., Sharathkumar, R.: Streaming algorithms for extent problems in high dimensions. *Algorithmica* **72**(1), 83–98 (2015)
6. Ahn, H.K., Kim, H.S., Kim, S.S., Son, W.: Computing  $k$  centers over streaming data for small  $k$ . *Int. J. Comput. Geom. Appl.* **24**(2), 107–123 (2014)
7. Ahn, H.K., Barba, L., Bose, P., Carufel, J.L.D., Korman, M., Oh, E.: A linear-time algorithm for the geodesic center of a simple polygon. *Discrete Comput. Geom.* **56**(4), 836–859 (2016). <https://doi.org/10.1007/s00454-016-9796-0>
8. Ahn, H.K., Kim, S.S., Knauer, C., Schlipf, L., Shin, C.S., Vigneron, A.: Covering and piercing disks with two centers. *Comput. Geom. Theor. Appl.* **46**(3), 253–262 (2013)
9. Bae, S.W.:  $L_1$  geodesic farthest neighbors in a simple polygon and related problems. *Discrete Comput. Geom.* **62**(4), 743–774 (2019)
10. Basappa, M., Jallu, R.K., Das, G.K.: Constrained  $k$ -center problem on a convex polygon. *Int. J. Found. Comput. Sci.* **31**(02), 275–291 (2020)
11. Chan, T.M.: More planar two-center algorithms. *Comput. Geom. Theor. Appl.* **13**(3), 189–198 (1999)
12. Chan, T., Pathak, V.: Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Comput. Geom. Theor. Appl.* **47**(2), 240–247 (2014)
13. Cho, K., Oh, E.: Optimal algorithm for the planar two-center problem. [arXiv:2007.08784](https://arxiv.org/abs/2007.08784) (2020)
14. Choi, J., Ahn, H.K.: Efficient planar two-center algorithms. *Comput. Geom.* **97**, 101768 (2021)
15. Choi, J., Jeong, D., Ahn, H.K.: Covering convex polygons by two congruent disks. [arXiv:2105.02483](https://arxiv.org/abs/2105.02483) (2021)
16. Cole, R.: Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM* **34**(1), 200–208 (1987)
17. Das, G.K., Roy, S., Das, S., Nandy, S.C.: Variations of base-station placement problem on the boundary of a convex region. *Int. J. Found. Comput. Sci.* **19**(02), 405–427 (2008)
18. Edelsbrunner, H., Kirkpatrick, D., Seidel, R.: On the shape of a set of points in the plane. *IEEE Trans. Inf. Theor.* **29**(4), 551–559 (1983)
19. Feder, T., Greene, D.: Optimal algorithms for approximate clustering. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pp. 434–444 (1988)
20. Fischer, K., Gärtner, B.: The smallest enclosing ball of balls: combinatorial structure and algorithms. *Int. J. Comput. Geom. Appl.* **4**(5), 341–378 (2004)
21. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.* **38**, 293–306 (1985)

22. Hershberger, J., Suri, S.: Adaptive sampling for geometric problems over data streams. *Comput. Geom. Theor. Appl.* **39**(3), 191–208 (2008)
23. Hershberger, J., Suri, S.: Finding tailored partitions. *J. Algorithms* **12**(3), 431–463 (1991)
24. Hwang, R., Lee, R.C.T., Chang, R.: The slab dividing approach to solve the Euclidean  $p$ -center problem. *Algorithmica* **9**(1), 1–22 (1993). <https://doi.org/10.1007/BF01185335>
25. Kim, S.S., Ahn, H.K.: An improved data stream algorithm for clustering. *Comput. Geom. Theor. Appl.* **48**(9), 635–645 (2015)
26. Kim, S.K., Shin, C.-S.: Efficient algorithms for two-center problems for a convex polygon. In: Du, D.-Z.-Z., Eades, P., Estivill-Castro, V., Lin, X., Sharma, A. (eds.) *COCOON 2000. LNCS*, vol. 1858, pp. 299–309. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-44968-X\\_30](https://doi.org/10.1007/3-540-44968-X_30)
27. Löffler, M., Van Kreveld, M.: Largest bounding box, smallest diameter, and related problems on imprecise points. *Comput. Geom. Theor. Appl.* **43**(4), 419–433 (2010)
28. Megiddo, N.: On the ball spanned by balls. *Discrete Comput. Geom.* **4**(6), 605–610 (1989). <https://doi.org/10.1007/BF02187750>
29. Megiddo, N.: Linear programming in linear time when the dimension is fixed. *J. ACM* **31**(1), 114–127 (1984)
30. Oh, E., De Carufel, J.L., Ahn, H.K.: The geodesic 2-center problem in a simple polygon. *Comput. Geom. Theor. Appl.* **74**, 21–37 (2018)
31. Roy, S., Bardhan, D., Das, S.: Base station placement on boundary of a convex polygon. *J. Parallel Distrib. Comput.* **68**(2), 265–273 (2008)
32. Sadhu, S., Roy, S., Nandi, S., Maheshwari, A., Nandy, S.C.: Two-center of the convex hull of a point set: dynamic model, and restricted streaming model. *Fundamenta Informaticae* **164**(1), 119–138 (2019)
33. Sharir, M.: A near-linear algorithm for the planar 2-center problem. *Discrete Comput. Geom.* **18**(2), 125–134 (1997)
34. Shin, C.-S., Kim, J.-H., Kim, S.K., Chwa, K.-Y.: Two-center problems for a convex polygon (extended abstract). In: Bilardi, G., Italiano, G.F., Pietracaprina, A., Pucci, G. (eds.) *ESA 1998. LNCS*, vol. 1461, pp. 199–210. Springer, Heidelberg (1998). [https://doi.org/10.1007/3-540-68530-8\\_17](https://doi.org/10.1007/3-540-68530-8_17)
35. Wang, H.: On the planar two-center problem and circular hulls. In: *Proceeding of the 36th International Symposium on Computational Geometry (SoCG 2020)*, vol. 164, pp. 68:1–68:14 (2020)
36. Wang, H.: On the planar two-center problem and circular hulls. *arXiv preprint [arXiv:2002.07945](https://arxiv.org/abs/2002.07945)* (2020)
37. Zarrabi-Zadeh, H.: Core-preserving algorithms. In: *Proceedings of the 20th Annual Canadian Conference on Computational Geometry (CCCG 2008)*, pp. 159–162 (2008)