Intersecting Disks using Two Congruent Disks*

Byeonguk Kang[1], Jongmin Choi[1], and Hee-Kap Ahn[2]

1 **Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea.**
{kbu417,icothos}@postech.ac.kr

2 **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 Euclidean 2-center problem for a set of n disks in the plane: find two smallest congruent disks such that every disk in the set intersects at least one of the two congruent disks. We present a deterministic algorithm for the problem that returns an optimal pair of congruent disks in $O(n^2 \log^3 n / \log\log n)$ time. We also present a randomized algorithm with $O(n^2 \log^2 n / \log\log n)$ expected time. These results improve the previously best deterministic and randomized algorithms, making a step closer to the optimal algorithms for the problem.

1 Introduction

We consider a generalization of the 2-center problem [1, 4, 7, 8] in which given a set $\mathcal{D}$ of n disks of nonnegative radii in the plane, find two smallest congruent disks C_1 and C_2 satisfying $D \cap (C_1 \cup C_2) \neq \emptyset$ for every $D \in \mathcal{D}$. We call this problem the *2-center problem on disks.*

Ahn et al. [2] gave a deterministic algorithm for the problem with $O(n^2 \log^4 n \log\log n)$ time and a randomized algorithm with $O(n^2 \log^3 n)$ expected time. They showed that their algorithms also work for the *restricted 2-cover problem* on disks (every disk is contained in one of two smallest congruent disks) and the *2-piercing problem on disks* (every disk is pierced by one of two optimal points) in the plane. See Figure 1 for an illustration.

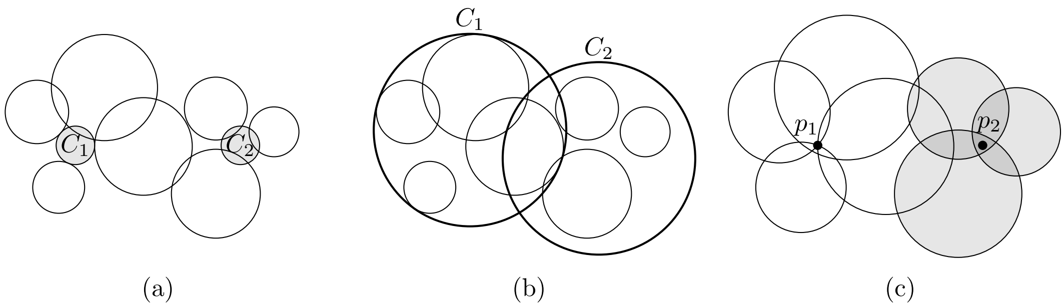

Figure 1 (a) The disk 2-center problem on disks: every input disk intersects $C_1 \cup C_2$. (b) The restricted disk 2-cover problem on disks: every disk is fully contained in C_1 or C_2. (c) The 2-piercing problem on disks: every disk intersects p_1 or p_2.

* This research was partly 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)).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Our results. We present a deterministic algorithm with $O(n^2 \log^3 n/\log\log n)$ time and a randomized algorithm with $O(n^2 \log^2 n/\log\log n)$ expected time for the 2-center problem and the restricted 2-cover problem on n disks in the plane. This improves the previously best known algorithm by more than $O(\log n)$ factor. Our deterministic algorithm also works on the 2-piercing problem with $O(n^2 \log^2 n/\log\log n)$ time.

For a disk D, the disk inflated by real value $r \geq 0$ from D, denoted by $D(r)$, is centered at the center of D and its radius is the radius of D plus r. We use a dual arrangement $\mathcal{A}$ of the disk centers and construct a dual directed tree $T_\mathcal{E}$ of $\mathcal{A}$. Our sequential decision algorithm traverses the tree in directions of inserting inflated disks one by one and finds the centers. Our sequential decision algorithm works as follows.

1. Construct a point-line dual arrangement $\mathcal{A}$ of the disk centers such that each face of $\mathcal{A}$ represents the inflated disks whose centers lie in one side of a line in primal space.
2. Construct a directed tree $T_\mathcal{E}$ such that there is a one-to-one correspondence between the tree nodes and the faces of $\mathcal{A}$, and each edge is directed from a node to a neighboring node of lower level in $\mathcal{A}$.
3. Construct a collection $\mathcal{T}_t$ of t-ary search trees that, given a face f in $\mathcal{A}$, returns $O(\log n/\log\log n)$ regions whose common intersection is the intersection of the inflated disks represented by f.
4. Check for each face f while traversing $T_\mathcal{E}$ if the inflated disks represented by f have a nonempty intersection and the remaining inflated disks also have a nonempty intersection, using $\mathcal{T}_t$ and an insertion-only convex programming.

Our deterministic algorithm uses Cole's parametric search [5] with an $O(n^2 \log^2 n/\log\log n)$-time sequential decision algorithm and an $O(\log n)$-time parallel decision algorithm using $O(n^2 \log^2 n/\log^2 \log n)$ processors, after $O(n^2 \log^3 n/\log\log n)$-time preprocessing. The improvement of the sequential decision algorithm comes from the insertion-only convex programming and the data structure $\mathcal{T}_t$. The parallel decision algorithm constructs $\mathcal{T}_t$ in the preprocessing phase and the convex programming runs in parallel. Putting them together using Cole's parametric search, we get an $O(n^2 \log^3 n/\log\log n)$-time algorithm. A randomized algorithm with $O(n^2 \log^2 n/\log\log n)$ expected time can be obtained by combining our sequential decision algorithm and Chan's randomized optimization technique [3].

2 Preliminaries

▶ **Observation 2.1** (Observation 1 in [2]). *Let (C_1, C_2) be a pair of optimal covering disks. Let ℓ be the bisector of the segment connecting the centers of C_1 and C_2. Then, $C_i \cap D \neq \emptyset$ for every $D \in \mathcal{D}$ whose center lies on the same side of ℓ as the center of C_i, for $i = \{1, 2\}$.*

For a line ℓ in the plane, let B_ℓ be a bipartition of $\mathcal{D}$ to $\mathcal{D}_\ell$ and $\mathcal{D}_\ell^c = \mathcal{D} \setminus \mathcal{D}_\ell$, where $\mathcal{D}_\ell$ is the set consisting of disks in $\mathcal{D}$ with centers lying strictly below ℓ. Based on Observation 2.1, B_ℓ defines a subproblem consisting of two 1-center problems such that the smallest radius for the subproblem is the larger one of the two radii from the 1-center problems.

Given a real value $r \geq 0$, the decision 2-center problem on $\mathcal{D}$ is to determine whether $r \geq r^*$, where r^* is the radius of the two smallest congruent disks of the 2-center problem on $\mathcal{D}$. Let $\mathcal{D}(r)$ be the set of the inflated disks $D(r)$ of disks $D \in \mathcal{D}$. Then the decision 2-center problem on $\mathcal{D}$ with radius r reduces to the 2-piercing problem on $\mathcal{D}(r)$.

Given compact convex subsets in the plane, each representing a constraint, and an objective function, a point that satisfies the constraints and minimizes the objective function value can be found using convex programming. There are two types of primitive operations:

finding the leftmost feasible point of two constraints, and determining whether a given point is contained in a constraint. Convex programming can be used to determine whether the intersection of input convex sets is empty or not.

▶ **Lemma 2.2.** *A problem with k convex constraints can be solved in $O(T_c + \log k)$ time using convex programming, with $O(k^2)$ processors, where T_c denotes the time per primitive operation.*

▶ **Lemma 2.3.** *Given a convex program with k constraints and the leftmost point v^* of the intersection of the constraints, the operation of adding a new constraint to the convex programming can be handled such that the leftmost point in the intersection of the $k+1$ constraints can be found in $O(kT_c)$ time, where T_c denotes the time per primitive operation.*

In the following, we consider the 2-piercing problem on inflated disks in $\mathcal{D}(r)$.

3 The 2-piercing Problem on Disks

We present an algorithm for the 2-piercing problem on a set $\mathcal{E} = \{E_1, E_2, \dots, E_n\}$ of n disks in the plane. For a disk set X, let $I(X)$ denote the intersection of the disks in X. If there is a bipartition B_ℓ such that both $I(\mathcal{E}_\ell)$ and $I(\mathcal{E}_\ell^c)$ are nonempty, the 2-piercing problem on $\mathcal{E}$ has the solution, where $\mathcal{E}_\ell$ is the set consisting of disks in $\mathcal{E}$ with centers lying strictly below ℓ, and $\mathcal{E}_\ell^c = \mathcal{E} \setminus \mathcal{E}_\ell$. For each bipartition B_ℓ, we perform the *emptiness test* which determines whether $I(\mathcal{E}_\ell) \neq \emptyset$ and $I(\mathcal{E}_\ell^c) \neq \emptyset$.

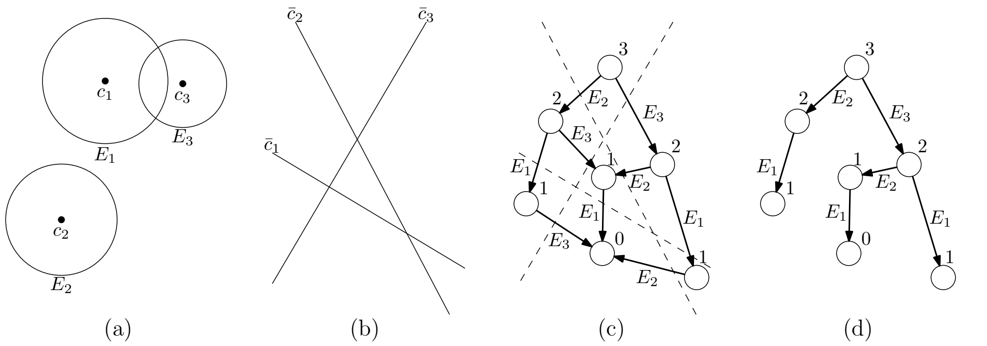

Figure 2 (a) Three disks E_1, E_2, E_3 with centers at c_1, c_2, c_3 in the plane. (b) Three dual lines $\bar{c}_1$, $\bar{c}_2$, $\bar{c}_3$ form the dual arrangement of the centers of disks in (a). (c) Dual graph $G_\mathcal{E}$ of the dual arrangement $\mathcal{A}$. The level of a node in $G_\mathcal{E}$ is the level of its corresponding face in $\mathcal{A}$. (d) Directed tree $T_\mathcal{E}$ from dual graph $G_\mathcal{E}$.

We construct the dual arrangement $\mathcal{A}$ for the centers of the disks in $\mathcal{E}$ by the following point-line duality transform: For a point $p := (p_x, p_y)$ in the primal plane, its dual $\bar{p}$ is the line $\bar{p} := (y = p_x x - p_y)$ in the dual plane. Likewise, for a line $\ell : y = \ell_x x + \ell_y$ in the primal plane, its dual $\bar{\ell}$ is the point $\bar{\ell} := (\ell_x, -\ell_y)$ in the dual plane. See Fig. 2(a,b). The duality transform preserves incidence ($p \in \ell$ if and only if $\bar{\ell} \in \bar{p}$) and order (p lies above ℓ if and only if $\bar{\ell}$ lies above $\bar{p}$) [6]. Thus, $\mathcal{A}$ is the arrangement induced by n lines in the dual plane, each of which is the dual of the center of an input disk. The level of a point in $\mathcal{A}$ is the number of lines in $\mathcal{A}$ lying on or below the point. For a face f of $\mathcal{A}$, let $\bar{\ell}$ be a point in f but not on the upper boundary chain of f. We define the level of f to be the level of $\bar{\ell}$. Let $\mathcal{E}_f$ denote the set of the disks in $\mathcal{E}$ such that the dual lines of their centers lie strictly above $\bar{\ell}$. Observe that $\mathcal{E}_f = \mathcal{E}_\ell$, and let $\mathcal{E}_f^c = \mathcal{E} \setminus \mathcal{E}_f$. Thus, they form the bipartition of the centers of input disks induced by ℓ in the primal plane.

3.1 Dual Directed Tree

Let $G_{\mathcal{E}}$ be a directed acyclic graph such that there is a one-to-one correspondence between the nodes of $G_{\mathcal{E}}$ and the faces in $\mathcal{A}$, and two nodes u, w of $G_{\mathcal{E}}$ are connected by a directed edge (u, w) from u to w if and only if the faces f_u and f_w corresponding to u and w, respectively, share a boundary edge and the level of f_u is larger than the level of f_w. There is a one-to-one correspondence between the bipartitions and the nodes of $G_{\mathcal{E}}$. For a node u in $G_{\mathcal{E}}$, let $\mathcal{E}_u = \mathcal{E}_f$ for face f of $\mathcal{A}$ corresponding to u, and let $\mathcal{E}_u^c = \mathcal{E} \setminus \mathcal{E}_u$. For each edge (u, w) of $G_{\mathcal{E}}$, $\mathcal{E}_w \setminus \mathcal{E}_u$ consists of exactly one disk, and (u, w) corresponds to the disk in $\mathcal{E}_w \setminus \mathcal{E}_u$. In Fig. 2(c), each directed edge is labeled with the disk corresponding to the edge.

Let v_r be the node of $G_{\mathcal{E}}$ that has no incoming edge. We construct from $G_{\mathcal{E}}$, a directed tree $T_{\mathcal{E}}$ rooted v_r that spans all vertices of $G_{\mathcal{E}}$, by choosing only one incoming edge for each node of $G_{\mathcal{E}}$. See Fig. 2(d). For two nodes u, w, let $p(u, w)$ denote the directed path from u to w in $T_{\mathcal{E}}$, if exists.

3.2 t-ary Search Trees

Let t be a parameter to be set later. For each leaf node v of $T_{\mathcal{E}}$, we construct a t-ary search tree $T_t(v)$ in bottom-up manner such that the leaf nodes are ordered from left to right, each corresponding to an edge in $p(v_r, v)$ in order from v_r to v, the leftmost t leaf nodes have the same parent node and the next t leaf nodes have the same parent node, and so on. This process goes recursively to higher levels, and $T_t(v)$ has height $h = O(\log_t n)$. See Fig. 3.

The path $p(v_r, v)$ represents a sequence of disks, each corresponding to an edge of the path. The data structure $T_t(v)$ supports queries that given a path $p(v_r, w)$ for a node w in $p(v_r, v)$, returns $h = O(\log_t n)$ subpaths that together form $p(v_r, w)$, and h intersections of disks, each corresponding to a subpath.

We construct a collection of t-ary search trees, one for each leaf node of $T_{\mathcal{E}}$, avoiding duplications of nodes as follows. First, we apply depth-first search (DFS) at v_r of $T_{\mathcal{E}}$, which gives us an order of the edges of $T_{\mathcal{E}}$, traversed by DFS. These edges are the leaf nodes of the collection, ordered from left to right following the order by DFS. Then we construct t-ary trees, in the order of the leaf nodes of $T_{\mathcal{E}}$ visited by DFS. For two leaf nodes v, v' with v visited before v', let v_{split} denote the lowest common ancestor node of $p(v_r, v)$ and $p(v_r, v')$. Then the path $p(v_r, v_{\text{split}})$ is the longest common subpath of the paths. To avoid duplications, $T_t(v')$ simply maintains a pointer to the part of $T_t(v)$ corresponding to $p(v_r, v_{\text{split}})$ with respect to t value, instead of constructing the part again. Let $\mathcal{T}_t$ denote the collection of all t-ary search trees. See Fig. 3(b) for an illustration.

3.3 Intersections of Disks for Paths

For a path $p(u, w)$, let $I(u, w)$ denote the intersection of the disks corresponding to $p(u, w)$. Observe that $I(v_r, w) = I(\mathcal{E}_w)$. For a node ν in $T_t(v)$, let ν^- be the left sibling node of ν, ν^+ the node next (right) to ν at the same level, and $\text{rc}(\nu)$ the rightmost child node of ν in $T_t(v)$. The leaf node ν of $T_t(v)$ corresponding to an edge e of $p(v_r, v)$ stores the intersection $I(\nu) = I(\nu^-) \cap D_e$ if ν^- is defined, and $I(\nu) = D_e$ otherwise, where D_e is the disk corresponding to e. A nonleaf node ν stores $I(\nu)$ if the subtree rooted at ν^+ is a perfect t-ary tree. We set $I(\nu) = I(\nu^-) \cap I(\text{rc}(\nu))$ if ν^- is defined, and $I(\nu) = I(\text{rc}(\nu))$ otherwise. See Fig. 3(b) for an illustration.

For a node ν, if $I(\nu)$ is stored at ν, $I(\nu) = I(w, w')$ for path $p(w, w')$ such that the edge of $p(w, w')$ incident to w corresponds to the leftmost leaf node in the subtree rooted at the

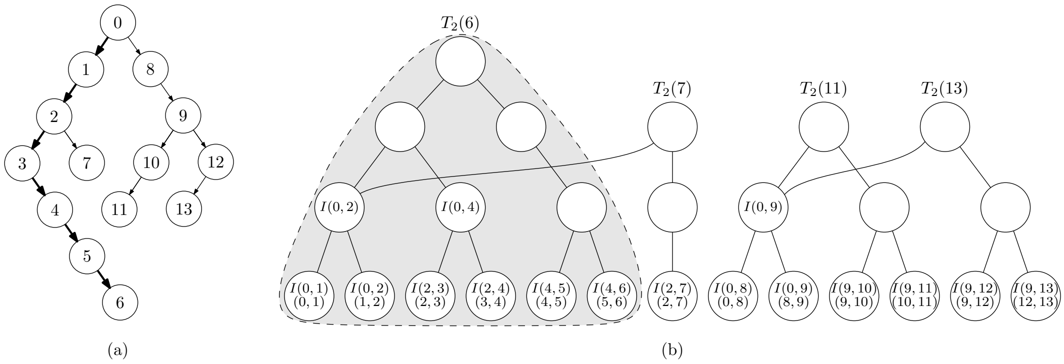

Figure 3 (a) Dual directed tree $T_{\mathcal{E}}$. (b) $\mathcal{T}_2$ for all leaf nodes in $T_{\mathcal{E}}$. $T_2(6)$ is constructed on $p(0,6)$ (thick path in (a)). Blank nodes in trees store no intersection of disks.

parent node of ν, and the edge of $p(w, w')$ incident to w' corresponds to the rightmost leaf node in the subtree rooted at ν. Thus, $I(\nu) = \bigcap D_e$ for all edges e in $p(w, w')$.

▶ **Lemma 3.1.** *Given a real value $r \geq 0$, we can construct $\mathcal{T}_t$ together with the intersections of disks stored at nodes in $O(tn^2 \log_t n)$ time using $O(tn^2 \log_t n)$ space.*

▶ **Lemma 3.2.** *Given a node w in $T_{\mathcal{E}}$, we can find a set $\mathcal{W}$ of $O(\log_t n)$ nodes in $\mathcal{T}_t$ such that $\cap_{\nu \in \mathcal{W}} I(\nu) = I(\mathcal{E}_w)$.*

3.4 Algorithm

If there is a node $u \in T_{\mathcal{E}}$ such that both $I(\mathcal{E}_u)$ and $I(\mathcal{E}_u^c)$ are nonempty, then the 2-piercing problem has a solution. Using $\mathcal{T}_t$ (Lemma 3.1 and Lemma 3.2), convex programming (Lemma 2.3) and setting $t = \log^\epsilon n$, we can solve the 2-piercing problem in $O(n^2 \log^2 n / \log \log n)$ time using $O(n^2 \log^{1+\epsilon} n)$ space for any constant $0 < \epsilon \leq 1$.

▶ **Theorem 3.3.** *Given a set of n disks in the plane, we can compute two points p_1 and p_2 such that every input disk contains p_1 or p_2 in $O(n^2 \log^2 n / \log \log n)$ time using $O(n^2 \log^{1+\epsilon} n)$ space for any constant $0 < \epsilon \leq 1$.*

4 The 2-center Problem on Disks

Our algorithms use parametric search which requires a sequential decision algorithm and a parallel decision algorithm.

4.1 Sequential Decision Algorithm

By solving the 2-piercing problem on the inflated disk in $\mathcal{D}(r)$, we can solve the decision 2-center problem with a given value r on $\mathcal{D}$.

▶ **Theorem 4.1.** *Given a set of n disks in the plane and a real value $r \geq 0$, we can determine whether there are two congruent disks C_1 and C_2 of radius r such that every input disk intersects C_1 or C_2 in $O(n^2 \log^2 n / \log \log n)$ time using $O(n^2 \log^{1+\epsilon} n)$ space for any constant ϵ with $0 < \epsilon \leq 1$.*

4.2 Parallel Decision Algorithm

We first describe a sequential preprocessing algorithm for finding an interval $(r_1, r_2]$ such that $r_1 < r^* \leq r_2$ and $\mathcal{T}_t$ has the same combinatorial structure for any $r \in (r_1, r_2]$, that is, for each intersection stored at nodes of $\mathcal{T}_t$, the circular arcs along the boundary are in the same order. The preprocessing consists of the construction of $\mathcal{T}_t$ for all $r \geq 0$ and binary search to find the interval $(r_1, r_2]$. To do this, we consider frustum F_i instead of $D_i(r)$ such that intersection of F_i and the plane $z = r$ is $D_i(r)$, for $i = 1, ..., n$.

▶ **Lemma 4.2.** *Given a set of n disks in the plane, we can construct $\mathcal{T}_t$ for all $r \geq 0$ in $O(tn^2 \log_t^2 n \cdot \log t)$ time. The space complexity of $\mathcal{T}_t$ for all $r \geq 0$ is $O(tn^2 \log_t n)$.*

▶ **Lemma 4.3.** *Given a set of n disks in the plane, we can find an interval $(r_1, r_2]$ in $O(n^2 \log^3 n / \log \log n)$ time such that $r_1 < r^* \leq r_2$ and $\mathcal{T}_t$ has the same combinatorial structure for any $r \in (r_1, r_2]$ and for $t = O(\log n)$.*

From the sequential preprocessing, we get $\mathcal{T}_t$ for an interval $(r_1, r_2]$ such that it has the same combinatorial structure for any $r \in (r_1, r_2]$ and $r^* \in (r_1, r_2]$. Using $\mathcal{T}_t$ and Lemma 2.2, we parallelize the process of determining $I(\mathcal{E}_u) = \emptyset$ and $I(\mathcal{E}_u^c) = \emptyset$ for all nodes $u \in T_{\mathcal{E}}$.

▶ **Theorem 4.4.** *Given a set of n disks in the plane and a real value $r \geq 0$, we can determine whether there are two congruent disks C_1 and C_2 of radius r such that every input disk intersects C_1 or C_2 in $O(\log n)$ time using $O(n^2 \log^2 n / \log^2 \log n)$ processors, after $O(n^2 \log^3 n / \log \log n)$-time preprocessing.*

4.3 Optimization Algorithms

We apply Cole's parametric search [5] to obtain an $O((P + T_s)(T_p + \log P))$-time deterministic algorithm, with our T_s-time sequential decision algorithm and our T_p-time parallel decision algorithm using P processors. Here $T_s = O(n^2 \log^2 n / \log \log n)$, $T_p = O(\log n)$ and $P = O(n^2 \log^2 n / \log^2 \log n)$. Thus, our deterministic algorithm runs in $O(n^2 \log^3 n / \log \log n)$ time. In addition, we apply Chan's randomized optimization [3] to obtain an $O(n^2 \log^2 n / \log \log n)$ expected time algorithm using our sequential decision algorithm.

▶ **Theorem 4.5.** *Given a set $\mathcal{D}$ of n disks in the plane, we can compute two smallest congruent disks C_1 and C_2 such that each disk in $\mathcal{D}$ intersects $C_1 \cup C_2$ in $O(n^2 \log^3 n / \log \log n)$ time. A randomized algorithm takes $O(n^2 \log^2 n / \log \log n)$ expected time.*

▶ **Corollary 4.6.** *Given a set of n disks in the plane, we can compute two smallest congruent disks C_1 and C_2 such that every disk is contained in either C_1 or C_2 in $O(n^2 \log^3 n / \log \log n)$ time. A randomized algorithm takes $O(n^2 \log^2 n / \log \log n)$ expected time.*

References

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

2 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*, 46(3):253–262, 2013.

3 T. M Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22(4):547–567, 1999.

4 T. M Chan. More planar two-center algorithms. *Computational Geometry*, 13(3):189–198, 1999.

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

6 M. de Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational geometry algorithms and applications*. Springer, 3rd edition, 2008.

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

8 H. Wang. On the planar two-center problem and circular hulls. In *Proceedings of the 36th International Symposium on Computational Geometry*, volume 164, pages 68:1–68:14, 2020.