Covering and Piercing Disks with Two Centers*

Hee-Kap Ahn† Christian Knauer‡ Sang-Sub Kim† Lena Schlipf§ Hyeon-Suk Na¶
Chan-Su Shin‖ Antoine Vigneron **

Abstract

We consider some variations of the two center problem. Let $\mathcal{D}$ denote a set of disks in the plane. We first study the problem of finding two smallest congruent disks such that each disk in $\mathcal{D}$ is intersected by one of these disks. Then we study the problem of covering the set $\mathcal{D}$ by two smallest congruent disks.

1 Introduction

There has been a fair amount of works on *the two center problem* for points: Given a set P of n points in the plane, find two smallest congruent disks that cover all points in P. So far, the best known algorithms run in $O(n \log^2 n \log^2 \log n)$ worst-case time [1] and in $O(n \log^2 n)$ expected time [4].

In this paper we consider two versions of the problem where the input consists of disks instead of points, so called *intersecting* and *covering* problems: Given a set $\mathcal{D}$ of n disks, compute two smallest congruent disks C_1 and C_2 such that each $D \in \mathcal{D}$ is intersected by C_1 or C_2 for the intersecting problem or is contained by the union of C_1 and C_2 for the covering problem.

The intersecting problem can be also formulated as a piercing problem: Compute the smallest value δ such that increasing the radius of every disk in $\mathcal{D}$ by δ makes the set pierceable by two points, meaning that there exist two points such that each disk contains at least one of the points. We present a simple $O(n^3)$ algorithm for this problem and algorithms which run in $O(n^2 \log^3 n)$ expected time and $O(n^2 \log^4 n \log \log n)$ time in the worst case.

The covering problems has also two different cases: In the *restricted case* each disk $D \in \mathcal{D}$ has to be fully covered by one of the disks C_1 or C_2. In the *general case* a disk $D \in \mathcal{D}$ can be covered by the union of C_1 and C_2. We show how the algorithms for the intersecting problem can be used to solve the restricted covering case. We complement these results by giving efficient approximation algorithms for the restricted and the general covering cases.

2 Preliminaries

The radius of a disks D is denoted by $r(D)$ and its center by $c(D)$. The distance between two disks D_1 and D_2 is denoted by $d(D_1, D_2)$ and is defined as $d(D_1, D_2) = d(c(D_1), c(D_2)) + r(D_1) + r(D_2)$.

3 Intersecting Disks with Two Disks

Given a set of disks $\mathcal{D} = \{D_1, \ldots, D_n\}$, we want to find two smallest congruent disks C_1 and C_2 such that for every disk $D_i \in \mathcal{D}$, D_i has a nonempty intersection with C_1 or C_2. Based on the observation below, we can design an algorithm that finds the optimal solution in $O(n^3)$ time.

Observation 1 *Let ℓ be the bisector of an optimal solution C_1 and C_2. Then, $C_i \cap D \neq \emptyset$ for any $D \in \mathcal{D}$ whose center lies in the same side of the center of C_i, for $i = \{1, 2\}$.*

For every bipartition of the centers of the disks in $\mathcal{D}$, we solve the 1-center problem for the disks in each partition. Then the center with larger radius is the solution for the 2-center problem restricted to the bipartition. We return the best one over all bipartitions. Since there are $O(n^2)$ such bipartitions and each 1-center problem can be solved in linear time [7], this algorithm runs in time $O(n^3)$.

To improve the time complexity, we formulate the problem in a different way as follows. For a real number δ, a *δ-inflated* disk of a disk D_i, denoted by $D_i(\delta)$,

*Work by Ahn and Kim was supported by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technology Development. Work by Schlipf was supported by the German Science Foundation (DFG) within the research training group 'Methods for Discrete Structures'(GRK 1408).

†Department of Computer Science and Engineering, POSTECH, Pohang, Korea. {heekap, helmet1981}@postech.ac.kr

‡Institute of Computer Science, Universität Bayreuth, 95440 Bayreuth, Germany. christian.knauer@uni-bayreuth.de

§Institute of Computer Science, Freie Universität Berlin, 14195 Berlin, Germany. schlipf@mi.fu-berlin.de

¶School of Computing, Soongsil University, Seoul, Korea. hsnaa@ssu.ac.kr

‖Department of Digital and Information Engineering, Hankuk University of Foreign Studies, Yongin, Korea. cssin@hufs.ac.kr

**Division of Mathematical and Computer Sciences and Engineering, KAUST, Thuwal, Saudi Arabia. antoine.vigneron@kaust.edu.sa

is a disk concentric to D_i with radius $r(D_i)+\delta$. Consider the following decision problem:

> Given a value δ, do there exist two points, p_1 and p_2, in the plane such that $D_i(\delta) \cap \{p_1, p_2\} \neq \emptyset$ for every $D_i \in \mathcal{D}$.

We let δ^* be the minimum value for which the decision problem answers "yes". To ensure that the radius of δ-*inflated* disks is nonnegative, we require that $\delta \geq -r_{\min}$, where $r_{\min}$ be the minimum of radii of all disks in $\mathcal{D}$. Without loss of generality, we also assume that no disk contains another disk in its closure, and that at most three δ-inflated disks intersect at a point (on their boundaries) for all disks in $\mathcal{D}$.

3.1 A decision algorithm

Given a value δ, we construct the arrangement of n δ-inflated disks in the plane. The arrangement consists of $O(n^2)$ cells. We traverse cells in the arrangement in depth-first manner and do the followings: We place one center point, say p_1, in a cell. The algorithm returns "yes" if all the disks that do not contain p_1 have a nonempty common intersection. Otherwise, we move p_1 to a neighboring cell, and repeat the test until we visit every cell. To test all cells in a naive way leads to a running time of $O(n^3)$.

To do the test efficiently, we partition the disks into $O(\sqrt{n})$ subsets with respect to their indices such that the first subset $\mathcal{D}_1$ consists of the first $\lceil\sqrt{n}\rceil$ disks $D_1, \cdots, D_{\lceil\sqrt{n}\rceil}$ and $\mathcal{D}_2$ consists of the following $\lceil\sqrt{n}\rceil$ disks, and so on. For each subset $\mathcal{D}_i$, we maintain a data structure representing the common intersection $I_i = \bigcap_{p \notin D, D \in \mathcal{D}_i} D$, of the disks in the set that do not contain p in its closure.

When we move p to a neighboring cell, at most one disk changes its status - therefore we update the common intersection of the corresponding group only, and this can be done in $O(\sqrt{n}\log n)$ time. If every I_i is nonempty, we decide whether they have a nonempty common intersection or not in $O(\sqrt{n}\log n)$ expected time using the randomized convex programming [3, 10] or in $O(\sqrt{n}\log^2 n)$ time using the deterministic convex programming [2]. Therefore the algorithm takes $O(n^{2.5}\log^2 n)$ time in total.

We can improve the time complexity further to $O(n^2 \log^2 n \log\log n)$ time or $O(n^2 \log^2 n)$ expected time by representing a traversal of cells as a set of intervals along a time line of the traversal, storing the set in segment trees, and checking the emptiness of the disk intersection in each cell with a help of the segment trees in $O(\log^2 n)$ expected time [3, 10] or $O(\log^2 n \log\log n)$ worst-case time [2]. Details will be found in the full version of the paper.

Lemma 1 *Given a value δ, we can decide in $O(n^2 \log^2 n \log\log n)$ time or in $O(n^2 \log^2 n)$ expected time whether there exist two points such that every δ-inflated disk is intersected by at least one of them.*

Note that the decision algorithm returns two solution points if the answer is "yes".

3.2 Finding δ^*

Lemma 2 *When $\delta = \delta^*$, either p_1 or p_2 is a common boundary point of three δ^*-inflated disks, a tangent point of two δ^*-inflated disks or a δ^*-inflated disk with radius zero.*

Thus, we consider only discrete values of δ for which one of the events defined in Lemma 2 occurs. If p_1 or p_2 is a δ-inflated disk with zero radius, that is, $\delta = -r_{\min}$, we can test whether the common intersection of the remaining δ-inflated disks is empty or not in $O(n)$ time. If p_1 or p_2 is a tangent point of two inflated disks, then we can collect such $O(n^2)$ tangent points (also associated radii) from all pairs of disks, and perform the binary search over the sorted radii by decision algorithm in Lemma 1 in $O(n^2 \log^3 n)$ expected time or $O(n^2 \log^3 n \log\log n)$ worst-case time. So from now on, we assume without loss of generality that p_1 is an intersection point of three δ-inflated disks.

For this case, we construct a frustum $f_i \in \mathbb{R}^3$ for each disk $D_i \in \mathcal{D}$. The bottom base of the frustum f_i is $D_i(-r_{\min})$ lying in the plane $z = -r_{\min}$. The intersection of f_i and the plane $z = \delta$ is $D_i(\delta)$. The top base of f_i is $D_i(\delta_{\max})$, where $\delta_{\max}$ is the radius of the smallest disk intersecting all disks in $\mathcal{D}$. Clearly, the optimal value of δ is in $[-r_{\min}, \delta_{\max}]$.

Let $p = (x, y)$ be the intersection point of the disks $D_i(\delta)$, $D_j(\delta)$, and $D_k(\delta)$. Then the point $p' = (x, y, \delta)$ is the intersection point of three frustums f_i, f_j, and f_k. Thus, p_1 is the projection of a point p' which is an intersection point of three frustums, i.e., a vertex of the arrangement $\mathcal{A}$ of the n frustums $f_1, \ldots f_n$; the complexity of $\mathcal{A}$ is $O(n^3)$. Note that for each candidate for p_1, the corresponding value for δ is easily obtained, namely the height of p'_1. Since $\mathcal{A}$ contains $O(n^3)$ vertices, we need to compute them implicity.

Implicit binary search We now describe how to perform the binary search over the vertices of $\mathcal{A}$ in an implicitly way:

Binary search on a coarse list of vertices. We first randomly select $O(n^2 \log n)$ vertices from $\mathcal{A}$ by picking $O(n^2 \log n)$ triples of frustums, and sort the radii associated with them in $O(n^2 \log^2 n)$ time. By a binary search with the decision algorithm in Lemma 1, we determine two consecutive radii δ_i and δ_{i+1} such that δ^* is between δ_i and δ_{i+1}. This takes $O(n^2 \log^3 n)$ time. Since the vertices were picked randomly, the strip $W[\delta_i, \delta_{i+1}]$ bounded by the two planes $z := \delta_i$ and $z := \delta_{i+1}$ contains only $O(n)$ vertices of $\mathcal{A}$ with high probability [9, Section 5].

Zooming into the interval. We compute all the k vertices in $W[\delta_i, \delta_{i+1}]$ by a standard sweep-plane algorithm in $O(k \log n + n^2 \log n)$ time as follows: First, we compute the intersection of the sweeping plane at $z := \delta_i$ with the frustums $f_1, \dots f_n$. This intersection forms a two-dimensional arrangement of $O(n)$ circles with $O(n^2)$ total complexity, and we can compute it in $O(n^2 \log n)$ time. We next construct the portion of the arrangement $\mathcal{A}$ in $W[\delta_i, \delta_{i+1}]$ incrementally by sweeping a vertical plane from the intersection at $z := \delta_i$ towards $z := \delta_{i+1}$. As a result, we can compute the $k = O(n)$ vertices (and the corresponding $O(n)$ radii) in $W[\delta_i, \delta_{i+1}]$ in $O(n \log n)$ time. We abort the sweep if the number k of vertices inside the strip becomes too large and restart the algorithm with a new random sample. This happens only with small probability. In order to find the minimum value δ^*, we perform a binary search on these $O(n)$ radii we just computed, using the decision algorithm in Lemma 1. This takes $O(n^2 \log^3 n)$ expected time. The solution pair of points p_1 and p_2 can also be found by the decision algorithm.

To get a deterministic algorithm, we use the parametric search technique with deterministic decision algorithm in Lemma 1, but the running time increases by one $(\log n)$-factor, so the total time becomes $O(n^2 \log^4 n \log\log n)$ time. Details can be found in the full version of this paper.

Theorem 3 *Given a set $\mathcal{D}$ of n disks in the plane, we can compute two smallest congruent disks whose union intersects every disk in $\mathcal{D}$ in $O(n^2 \log^3 n)$ expected time or in $O(n^2 \log^4 n \log\log n)$ worst-case time.*

4 Covering Disks with Two Disks

Given a set of disks $\mathcal{D} = \{D_1, \dots, D_n\}$ in the plane, we want to find two smallest congruent disks C_1 and C_2, such that every disk $D_i \in \mathcal{D}$ is covered by C_1 or C_2. As mentioned in the introduction, we distinguish between two cases: The *restricted case* where each disk D_i has to be fully covered by C_1 or C_2 and the *general case* where a disk D_i can be covered by $C_1 \cup C_2$.

4.1 The Restricted Case

Observation 2 *Let ℓ be the bisector of an optimal solution C_1 and C_2. Then, $D \subset C_i$ for every $D \in \mathcal{D}$ whose center lies in the same side of the center of C_i, for $i = \{1, 2\}$.*

Hence, the restricted covering case can be solved in $O(n^3)$ time, since the smallest disk covering a set of disks can be computed in linear time [8] and there are $O(n^2)$ bipartitions of the centers of the disks.

The algorithm from Section 3 can be also be used to solve this problem. For this we consider the decision problem: Given a set of n disks $\mathcal{D}$ and a value δ, do there exist two disks C_1, C_2 with radius δ, such that each disk $D_i \in \mathcal{D}$ is covered by either C_1 or C_2. This implies that for each disk $D_j \in \mathcal{D}$ covered by C_i holds: $d(c(D_j), c(C_i)) + r(D_j) \leq \delta$, for $i = \{1, 2\}$. It clearly holds that $\delta \geq r_{\max}$, where $r_{\max}$ is the maximum of radii of all disks in $\mathcal{D}$. We can formulated the problem in a different way.

> Given a value δ, do there exist two points, p_1 and p_2, such that $D_i^*(\delta) \cap \{p_1, p_2\} \neq \emptyset$ for every $D_i \in \mathcal{D}$, where $D_i^*(\delta)$ is a disk concentric to D_i and whose radius is $\delta - r(D_i) \geq 0$.

Since $\delta \geq r_{\max}$, we add an initialization step, in which every disk D_i is replaced by a disk concentric to D_i with radius $r_{\max} - r(D_i)$. Then we can use the algorithm from Section 3 in order to solve the restricted covering problem.

Theorem 4 *Given a set of n disks $\mathcal{D}$ in the plane, we can compute two smallest congruent disks such that each disk in $\mathcal{D}$ is covered by one of the disks in $O(n^2 \log^3 n)$ expected time or in $O(n^2 \log^4 n \log\log n)$ worst-case time.*

Constant factor approximation Algorithm 1 computes a $2/\sqrt{3}$-approximation in $O(n \log n)$ time. OneCover($\mathcal{U}$), which is used as a subroutine, computes the smallest disk covering a set of disks $\mathcal{U}$. Algorithm 1 runs in $O(n \log n)$ time, since the diameter of $\mathcal{D}$ can be computed in $O(n \log n)$ [7] and OneCover($\mathcal{U}$) in linear time [8]. The approximation factor of $2/\sqrt{3}$ can be proven by making a case distinction whether D_1 and D_2 are covered by the same disk in the optimal solution or by different disks, and by using Jung's theorem [5].

Algorithm 1

```
1: U1 = ∅ and U2 = ∅.
2: Compute the diametral pair D1 and D2, that is,
   d(D1, D2) is the diameter of D.
3: U1 = U1 ∪ {D1} and U2 = U2 ∪ {D2}.
4: for all Di ∈ D do
5:     if d(Di, D1) < d(Di, D2) then
6:         U1 = U1 ∪ {Di}
7:     else
8:         U2 = U2 ∪ {Di}
9: Compute C1 =OneCover(U1) and C2 =OneCover(U2)
10: return C1 and C2
```

At the expense of increasing the approximation factor by a factor of $\sqrt{2}$, we can improve the running time to $O(n)$ by replacing the computation of the diameter of $\mathcal{D}$ in Algorithm 1 by computing a $1/\sqrt{2}$-approximation of the diameter in $O(n)$ time.

Theorem 5 *Given a set of n disks $\mathcal{D}$. For the restricted covering case a $2/\sqrt{3}$-approximation can be computed in $O(n \log n)$ time and a $2\sqrt{2}/\sqrt{3}$-approximation in $O(n)$ time.*

$(1+\epsilon)$-**approximation** Recall Observation 2. We show how to compute an optimal solution in $O(n \log n)$ time if the orientation of the bisector is given and explain how this algorithm is used in order to obtain a $(1+\epsilon)$ approximation.

Fixed Orientation. W.l.o.g, assume that the bisector is vertical. After sorting the centers of all $D_i \in \mathcal{D}$ by their x-values, we sweep a vertical line ℓ from left to right, and maintain two sets $\mathcal{D}_1$ and $\mathcal{D}_2$: $\mathcal{D}_1$ contains all disks whose centers lie to the left of ℓ and $\mathcal{D}_2 = \mathcal{D} \setminus \mathcal{D}_1$. Let C_1 be the smallest disk covering $\mathcal{D}_1$ and C_2 the smallest disk covering $\mathcal{D}_2$. While sweeping ℓ from left to right, the radius of C_1 is nondecreasing and the radius of C_2 nonincreasing and we want to compute $\min \max(r(C_1), r(C_2))$. Hence, we can perform a binary search on the list of all centers. Each step takes $O(n)$ time, thus we achieve a total running time of $O(n \log n)$.

Sampling. We use $2\pi/\epsilon$ sample orientations chosen regularly over 2π, and compute for each orientation the solution in $O(n \log n)$ time. The approximation factor can be proven by showing that there is a sample orientation that makes angle at most ϵ with the optimal bisector. The solution for this line is at most $(1+\epsilon)$ times the optimal solution.

Theorem 6 *Given a set $\mathcal{D}$ of n disks in the plane, a $(1+\epsilon)$ approximation for the restricted covering problem for $\mathcal{D}$ can be computed in $O(n \log n/\epsilon)$ time.*

4.2 The General Case

Lemma 7 *A solution for the restricted case is a $(\sqrt{2}+1)/2$-approximation for the general case.*

Theorem 8 *Given a set $\mathcal{D}$ of n disks, a $(\sqrt{2}+1)/\sqrt{3}$-approximation for the general covering problem for $\mathcal{D}$ can be computed in $O(n \log n)$ time and a $(2+\sqrt{2})/\sqrt{3}$-approximation in $O(n)$ time.*

$(1+\epsilon)$-**Approximation** First, we compute a $(2+\sqrt{2})/\sqrt{3}$-approximation for the general covering case in $O(n)$ time as explained before. Let C_1^{apx}, C_2^{apx} be the solution disks. We can assume that $d(c(C_1^{\text{apx}}), c(C_2^{\text{apx}})) \leq (2+2(2+\sqrt{2})/\sqrt{3}) r(C_1^{\text{apx}}) < 6r(C_1^{\text{apx}})$, otherwise C_1^{apx}, C_2^{apx} are already an optimal solution. We compute a smallest bounding box B of $C_1^{\text{apx}} \cup C_2^{\text{apx}}$ and an $O(1/\epsilon) \times O(1/\epsilon)$ grid on B. The length of each grid cell is $< 6\epsilon r(C_1^{\text{apx}}) < 12\epsilon r^*$, where r^* is the radius of the optimal solution disks. Each disk is replaced in $O(n/\epsilon)$ time by the grid points which are closest to its boundary and lie inside the disk. If a disk has no grid point lying inside, it is replaced by the grid point which is closest to this disk. In total we have a set of $O(1/\epsilon^2)$ points for which we solve the two center problem, meaning we compute the smallest two disks E_1, E_2 that cover this point set in $O(\frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon} \log^2 \log \frac{1}{\epsilon})$ time [1]. It holds that $r(E_1) = r(E_2) \leq r^*$ and if the radii of E_1, E_2 are increased by the length of the diagonal of a grid cell, which is $< 6\epsilon\sqrt{2} r(C_1^{\text{apx}})$, these disks cover $\mathcal{D}$ and their radii are $< r^*(1+12\sqrt{2}\epsilon)$. The total running time is $O(\frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon} \log^2 \log \frac{1}{\epsilon} + \frac{n}{\epsilon})$.

In order to improve the running time for very small ϵ, i.e., $\epsilon \leq 1/\log n$, we compute the set of grid points from the union of the disks in $\mathcal{D}$. The union can be computed in $O(n \log n)$ time and its complexity is $O(n)$ [6]. Hence, the replacement of the disks by grid points takes $O(n \log n + 1/\epsilon^2)$ time.

Theorem 9 *Given a set $\mathcal{D}$ of n disks in the plane, a $(1+\epsilon)$-approximation for $\mathcal{D}$ in the general covering case can be computed in $O(\frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon} \log^2 \log \frac{1}{\epsilon} + \min\{\frac{n}{\epsilon}, n \log n\})$ time.*

References

[1] T. M. Chan. More planar two-center algorithms. *Comput. Geom. Theory Appl.*, 13:189–198, 1997.

[2] T. M. Chan. Deterministic algorithms for 2-d convex programming and 3-d online linear programming. *J. Algorithms*, 27(1):147–166, 1998.

[3] K. L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[4] D. Eppstein. Faster construction of planar two-centers. In *Proc. SODA'97*, pages 131–138. ACM and SIAM, 1997.

[5] H. Jung. Über den kleinsten Kreis, der eine ebene Figur einschließt. *J. Reine Angew. Math.*, 137:310–313, 1910.

[6] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[7] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Comput. Geom. Theory Appl.*, 43:419–433, 2010.

[8] N. Megiddo. On the ball spanned by balls. *Discrete Comput. Geom.*, 4:605–610, 1989.

[9] K. Mulmuley. *Computational Geometry - An Introduction Through Randomized Algorithms.* Prentice Hall, 1994.

[10] M. Sharir and E. Welzl. A Combinatorial Bound for Linear Programming and Related Problems. In *Proc. STACS'92*, pages 569–579. Springer-Verlag, 1992.