Improved Local Search for Geometric Hitting Set

Norbert Bus, Shashwat Garg, Nabil Mustafa, Saurabh Ray

Improved Local Search for Geometric Hitting Set

Norbert Bus[1], Shashwat Garg[2], Nabil Mustafa[3], and Saurabh Ray[4]

1 **Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge.** busn@esiee.fr
2 **Indian Institute of Technology, Delhi.** garg.shashwat@gmail.com
3 **Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge.** mustafan@esiee.fr
4 **Computer Science Department, New York University, Abu Dhabi.** saurabh.ray@nyu.edu

Abstract

Over the past several decades there has been steady progress towards the goal of polynomial-time approximation schemes (PTAS) for fundamental geometric combinatorial optimization problems. A foremost example is the geometric hitting set problem: given a set P of points and a set $\mathcal{D}$ of geometric objects, compute the minimum-sized subset of P that hits all objects in $\mathcal{D}$. For the case where $\mathcal{D}$ is a set of disks in the plane, the 30-year quest for a PTAS, starting from the seminal work of Hochbaum [19], was finally achieved in 2010. Surprisingly, the algorithm to achieve the PTAS is simple: local-search. Since then, local-search has turned out to be a powerful algorithmic approach towards achieving good approximation ratios for geometric problems (for geometric independent-set problem, for dominating sets, for the terrain guarding problem and several others).

Unfortunately all these algorithms have the same limitation: local search is able to give a PTAS, but with hopeless running times; e.g., a 3-approximation for the geometric hitting takes more than n^{66} time [15] for the geometric hitting set problem for disks in the plane! That leaves open the question of whether a better understanding – both combinatorial and algorithmic – of local search and the problem can give a better approximation ratio in a more reasonable time. In this paper, we investigate this question for one of the fundamental problems, hitting sets for disks in the plane. We present tight approximation bounds for $(3, 2)$-local search[1] and give an $(8 + \epsilon)$-approximation algorithm with running time $O(n^{2.34})$; the previous-best result achieving a similar approximation ratio gave a 10-approximation in time $O(n^{15})$ – that too just for unit disks. The techniques and ideas generalize to $(4, 3)$ local search. Furthermore, as mentioned earlier, local-search has been used for several other geometric optimization problems; for all these problems our results show that $(3, 2)$ local search gives an 8-approximation and no better [2]. Similarly $(4, 3)$-local search gives a 5-approximation for all these problems.

1 Introduction

The minimum hitting set problem is one the most fundamental combinatorial optimization problems: given a range space $(P, \mathcal{D})$ consisting of a set P and a set $\mathcal{D}$ of subsets of P called the *ranges*, the task is to compute the smallest subset $S \subseteq P$ that has a non-empty intersection with each of the ranges in $\mathcal{D}$. This problem is strongly NP-hard. If there are no restrictions on the set system $\mathcal{D}$, then it is known that it is NP-hard to approximate the minimum hitting set within a logarithmic factor of the optimal [25]. A natural occurrence of the hitting set problem occurs when the range space $\mathcal{D}$ is derived from geometry. For example, given a set P of n points in $\mathbb{R}^2$, and a set $\mathcal{D}$ of m triangles containing points of P, compute the minimum-sized subset of P that hits all the triangles in $\mathcal{D}$.

[1] Note that a $(2, 1)$-local search does not give any bounded approximation ratio.
[2] This is assuming the use of the standard framework. Improvement of the approximation factor by using additional properties specific to the problem may be possible.

Unfortunately, for most natural geometric range spaces, computing the minimum-sized hitting set remains NP-hard. For example, even the (relatively) simple case where $\mathcal{D}$ is a set of unit disks in the plane is strongly NP-hard [18]. Therefore fast algorithms for computing provably good approximate hitting sets for geometric range spaces have been intensively studied for the past three decades (e.g., see the two recent PhD theses on this topic [15, 16]). Since there is little hope of computing the minimum-sized hitting set for general geometric problems in polynomial time, effort has turned to approximating the optimal solution.

In this paper we will consider a fundamental case for geometric hitting sets, where the geometric objects are arbitrary radius disks in the plane (or halfspaces in $\mathbb{R}^3$). This has been the subject of a long line of investigation, for more than two decades. The case when all the disks have the same radius is easier, and has been studied in a series of works: Călinsecu *et al.* [9] proposed a 108-approximation algorithm, which was subsequently improved by Ambhul *et al.* [6] to 72. Carmi *et al.* [10] further improved that to a 38-approximation algorithm, though with the running time of $O(n^6)$. Claude *et al.* [11] were able to achieve a 22-approximation algorithm running in time $O(n^6)$. More recently Fraser *et al.* [17] presented a 18-approximation algorithm in time $O(n^2)$.

The problem becomes harder when the disk radii can be arbitrary. A first break-through for this problem came in 1994, when Bronnimann and Goodrich [8] proved the following interesting connection between the hitting-set problem, and ϵ-nets[3]: let $(P, \mathcal{D})$ be a range-space for which we want to compute a minimum hitting set. If one can compute an ϵ-net of size c/ϵ for the ϵ-net problem for $(P, \mathcal{D})$ in polynomial time, then one can compute a hitting set of size at most $c \cdot \text{OPT}$ for $(P, \mathcal{D})$, where OPT is the size of the optimal (smallest) hitting set, in polynomial time. A shorter, simpler proof was given by Even *et al.* [13]. Recently, Agarwal-Pan [5] presented an algorithm that can compute hitting-sets for disks from ϵ-nets in time $O(n \log^6 n)$, improving on the previous near-linear time algorithm which guaranteed a $O(\log n)$-approximation [3].

Local search.

There is a fundamental limitation of the above technique: it *cannot* give better than constant-factor approximations. The reason is that the technique reduces the problem of computing a minimum size hitting set to the problem of computing a minimum sized ϵ-net. And it is known that for some constant $c \geq 2$, there do not exist ϵ-nets of size smaller than c/ϵ – even for halfspaces in 2D. This limitation was the main barrier towards better quality algorithms until the usefulness of local search algorithms was introduced.

There has been recent progress in breaking the constant-approximation barriers for many geometric problems using very similar applications of local-search; e.g., dominating sets in disk intersection graphs, terrain guarding problem, independent set of pseudodisks and several other problems. For the hitting set problem on $(P, \mathcal{D})$, local-search works as follows: start with any hitting set $S \subseteq P$, and repeatedly decrease the size of S, if possible, by replacing k points of S with $< k$ points of $P \setminus S$. Call such an algorithm a $(k, k-1)$-local search algorithm. Mustafa-Ray [21] showed that a $(k, k-1)$-local search algorithm for the hitting set problem gives a $(1 + c/\sqrt{k})$-approximation, for a fixed constant c, when the ranges are disks, or more generally, pseudo-disks in $\mathbb{R}^2$. The running time of their algorithm to compute a $(1+\epsilon)$-approximation is $O(n^{O(1/\epsilon^2)})$.

Our Contributions.

Both these approaches have to be evaluated on the questions of computational efficiency as well as approximation quality. In spite of all the progress, there remains a large gap between quality and

[3] Given a range space $(P, \mathcal{D})$, an ϵ-net is a subset $S \subseteq P$ such that $D \cap S \neq \emptyset$ for all $D \in \mathcal{D}$ with $|D \cap P| \geq \epsilon n$.

CONGRUENT DISKS		
	Quality	**Time**
Călinsecu *et al.* [9]	108	$O(n^2)$
Ambhul *et al.* [6]	72	$O(n^2)$
Carmi *et al.* [10]	38	$O(n^6)$
Claude *et al.* [11]	22	$O(n^6)$
Fraser *et al.* [17]	18	$O(n^2)$
Acharyya *et al.* [1]	$(9+\epsilon)$	$O(n^{2(1+6/\epsilon)+1)})$

ARBITRARY DISKS		
	Quality	**Time**
Bronniman *et al.* [8]	$O(1)$	$O(n^3)$
Mustafa *et al.* [21]	$(1+\epsilon)$	$n^{O(1/\epsilon^2)}$
Agarwal *et al.* [3]	$O(\log n)$	$\tilde{O}(n)$
Agarwal *et al.* [5]	$O(1)$	$\tilde{O}(n)$
This paper	$8+\epsilon$	$\tilde{O}(n^{7/3})$
This paper	$5+\epsilon$	$\tilde{O}(n^{3.75})$

Table 1 Summary of previous work.

efficiency – mainly due to the ugly trade-offs between running times and approximation factors; see Table 1 for the current state of the art. The algorithm of Agarwal-Pan [5] that rounds via ϵ-nets gives an $\tilde{O}(n)$-time algorithm, but the constant in the approximation depends on the constant in the size of ϵ-nets, which is large. For disks in the plane, the current best size of ϵ-net is at least $40/\epsilon$ [24], yielding at best a 40-approximation algorithm. At the other end, the $(k, k-1)$-local search algorithm [21] can compute solutions arbitrarily close to the optimal, but it is extremely inefficient, even for reasonable approximation factors. For example, it takes time $O(n^{66})$[15] to compute a 3-approximation. Furthermore, note that any attempts at progress on improving local search must take into account that as the hitting set problem is strongly NP-hard, it is unlikely that algorithms exist that do not have a dependency on $1/\epsilon$ in the exponent.

Therefore in this paper we undertake a closer study of $(k, k-1)$-local search for small values of k. Table 1 states our contributions. As our first result, we determine the exact limits of $(3, 2)$-local search:

▶ Theorem (Proof in Section 2). A $(3, 2)$-local search algorithm returns a 8-approximation to the minimum hitting set. Furthermore, there exist a set P of points and a set $\mathcal{D}$ of disks where $(3, 2)$ local-search does not return hitting-sets of size less than 8 factor of the optimal hitting set.

Remark: In fact this immediately implies improved bounds for many other local search algorithms; e.g., it implies that the $(3, 2)$-local search gives 8-approximation to the independent set of pseudodisks, dominating sets in disk intersection graphs, terrain guarding problem. We leave the details of these further applications to the full paper.

A straightforward algorithm for $(3, 2)$-local search proceeds as follows: each $(3, 2)$ improvement step tries all $O(n^5)$ 5-tuples, and for each checks if it is indeed an improvment in time $O(n)$. The total number of steps for the whole algorithm can be $O(n)$, giving a $O(n^7)$ naive running time. We show how to perform this search more efficiently:

▶ Theorem (Proof in Section 3). A $(3, 2)$-local search can be performed in expected time $O(n^{2.34})$.

In fact, these techniques can be generalized for larger values of k. For example, it can be shown that $(4, 3)$-local search gives a 5-approximation in time $\tilde{O}(n^{3.75})$. As the details are similar, we leave the proof for the full version of the paper.

2 Analysis of Quality for Local Search

Let R be a region in the plane. We say that a point $p \in \mathbb{R}^2$ hits R if $p \in R$, and that a set of points X hits a set of regions $\mathcal{R}$ if each region in $\mathcal{R}$ is hit by some point in X. We denote by $\mathcal{H}(P, \mathcal{R})$ the

set system $(P, \{R \cap P : R \in \mathcal{R}\})$ induced by P and $\mathcal{R}$. A hitting set for $\mathcal{H}(P, \mathcal{R})$ is a subset of P which hits $\mathcal{R}$. A hitting set of the smallest cardinality is called the minimum hitting set and its size is denoted $\textsc{Opt}(P, \mathcal{R})$ (or simply $\textsc{Opt}$ when it is clear from the context). From now onwards, P denotes a set of points and $\mathcal{D}$ denotes a set of (circular) disks in the plane. Our goal is to compute a hitting set for $\mathcal{H}(P, \mathcal{D})$ of a small size efficiently.

The analysis of the approximation factor achieved by a $(k, k-1)$-local search depends on the following theorem on planar bipartite graphs.

▶ **Theorem 1.** *[21] Let $G = (R, B, E)$ be a bipartite planar graph on red and blue vertex sets R and B, such that for every subset $B' \subseteq B$ of size at most k, where k is a large enough number, $|N_G(B')| \geq |B'|$. Then, $|B| \leq (1 + c/\sqrt{k})\,|R|$, where c is a constant.*

Here $N_G(B')$ denotes the set of neighbors of the vertices in B' in G. The proof of the above theorem, which relies on planar graph separators, requires k to be quite large, thereby limiting the practical utility of the above theorem. A priori, it is not clear whether the theorems holds at all for small values of k. For instance, one can easily see that for $k = 2$ there is no upper bound on $|B|/|R|$ (e.g., consider complete bipartite graph where B is arbitrarily large and $|R| = 2$). However, for $k = 3$, we show a small bound of 8 on $|B|/|R|$, and then prove that it is, in fact, optimal.

▶ **Theorem 2.** *Let $G = (R, B, E)$ be a bipartite planar graph on red and blue vertex sets R and B, such that for every subset $B' \subseteq B$ of size at most 3, $|N_G(B')| \geq |B'|$. Then, $|B| \leq 8\,|R|$ and this bound is tight.*

Proof. Let $n_b = |B|$ and $n_r = |R|$. Our goal is to prove that $n_b \leq 8n_r$. Note that no vertex in B can have degree 0, otherwise the neighborhood of such a vertex is of size 0, violating the conditions of the theorem. We make a new graph G' by adding edges in G to all vertices of B which have degree 1 in G. This can always be done while maintaining the planarity and bipartiteness of the graph as any such vertex v must lie in a face which has at least two vertices of R, at least one of which is not adjacent to v. Thus in G' every vertex in B has degree at least 2. Let n_{b_2} be the number of vertices of B which have degree 2 and $n_{b_{\geq 3}} = n_b - n_{b_2}$ be the number of vertices of B which have degree at least 3 in G'. Since G' is planar and bipartite the number of edges in $G' \leq 2(n_b + n_r)$. This implies that $2n_{b_2} + 3n_{b_{\geq 3}} \leq 2n_b + 2n_r$. Since $n_b = n_{b_2} + n_{b_{\geq 3}}$, we obtain $n_{b_{\geq 3}} \leq 2n_r$.

We now show that $n_{b_2} \leq 6n_r$. To do that we construct a graph H with vertex set R as follows: two vertices $r_1 \in R$ and $r_2 \in R$ are adjacent in H iff there is at least one vertex $b \in B$ of degree 2 which is adjacent to both r_1 and r_2 in G'. Note that H is planar since the edge between r_1 and r_2 can be routed via one such b. Note that for the same pair $\{r_1, r_2\}$ there cannot be three vertices $b_1, b_2, b_3 \in B$ of degree 2 each that are adjacent to both r_1 and r_2 since in that case the neighborhood of the set $\{b_1, b_2, b_3\}$ is of size 2 violating the conditions of the theorem. Therefore, each vertex $b \in B$ of degree 2 corresponds to an edge in H and each edge has at most two vertices in B that correspond to it. Since the number of edges in H is at most $3|R| = 3n_r$, we conclude that $n_{b_2} \leq 6n_r$. Thus $n_b = n_{b_2} + n_{b_{\geq 3}} \leq 6n_r + 2n_r = 8n_r$. ◀

We now show that the bound given above is tight. However, that still leaves open the possibility that, by exploiting other properties of disks, a $(3, 2)$-local search could give a better approximation for the problem of computing minimum hitting sets for disks in the plane. The following theorem rules this out.

▶ **Theorem 3.** *For any $\delta > 0$, there exists an integer n_0 such that one can construct a set $\mathcal{D}$ of disks in the plane, a set of points P, $|P| \geq n_0$, and a subset $B \subseteq P$ s.t. i)B is a hitting set for $\mathcal{H}(P, \mathcal{D})$,*

ii) $|B| \geq (8-\delta)\textsc{Opt}$ *and iii) there are no subsets* $X \subseteq B$ *and* $Y \subseteq P \setminus B$, $|Y| < |X| \leq 3$, *s.t.* $(B \setminus X) \cup Y$ *is a hitting set for* $\mathcal{H}(P, \mathcal{D})$.

Proof. We first construct a bipartite graph $G = (R, B, E)$ that satisfies the conditions of Theorem 2 and $|B| \geq (8-\delta)|R|$. Let L be the triangular lattice, and take a large equilateral triangle Δ aligned with the edges of L (so that $L \cap \Delta$ triangulates Δ) and containing many faces of the lattice. Then replace each face of the lattice by the block of the type shown in Figure 1(a). The corner vertices (unshaded) of the block map to the corner vertices of the face, while the other vertices (shaded) in the block lie in the interior of the face. Let R be the set of vertices of L lying in Δ and let B be the set of vertices lying in the interior of the faces in $L \cap \Delta$. The blocks together define a bipartite graph (see Figure 1(b) for a small example with four blocks put together; note that the edges of L are drawn in bold). The dotted edges and the edges of the lattice L are not part of the graph. Note that each face in $L \cap \Delta$ contains four points of B, and if Δ is large enough, the number of faces of L in Δ is nearly twice the number of vertices of L in Δ. The size of Δ can be chosen, depending on δ, such that the number of faces of L is at least $(2 - \delta/4)$ times the number of vertices of L. Thus we get that $|B| \geq (8-\delta)|R|$. It can be verified by inspection that there is no subset of B of size at most 3 with a smaller neighborhood. This shows that the bound in Theorem 2 is tight within additive constants.

Now, we extend G to a triangulation by including the dotted edges in the blocks. Note that there are some dotted edges going between blocks. We also put an additional vertex in the outer face and connect it to all vertices in the outer face of G (i.e. we stellate the outer face). The resulting graph, call it Ξ, is triangulated (i.e., each face is of size 3) and furthermore it is 4-connected since, as can be verified by inspection, there is no separating triangle (a non-facial cycle of length 3). By a theorem of Dillencourt and Smith (Theorem 3.5 in [12]), there exists an embedding of Ξ in the plane so that Ξ is the Delaunay triangulation of its vertices. Abusing notation, we refer to the embedding as Ξ and we refer to the embedding of a vertex v in Ξ as v.

Let R and B thus be the two sets of points. We set $P = R \cup B$, and construct $\mathcal{D}$ by taking for each edge e in G a disk that contains exactly the two end points of e among all the vertices in Ξ. This is possible because Ξ is now a Delaunay triangulation of the points in P. By construction, each disk in $\mathcal{D}$ contains exactly one point from each of the sets R and B and thus both the sets are hitting sets for $\mathcal{H}(P, \mathcal{D})$. Since $\textsc{Opt}$ is the size of the smallest hitting set, $\textsc{Opt} \leq |R|$ and therefore $|B| \geq (8-\delta)\textsc{Opt}$. Consider a local improvement step where we seek to decrease the size of the hitting set B by removing some subset $X \subseteq B$ of size at most 3 and adding a smaller set Y outside B (i.e., $Y \subseteq R$) so that $(B \setminus X) \cup Y$ is a hitting set for $\mathcal{D}$. Let x be one of the points in X. Observe that then all neighbors of x in G must be in Y since for each neighbor y of x, there is a disk in $\mathcal{D}$ which contains only the two points x and y among all the points in $R \cup B$. This means that $|Y| \geq |N_G(X)|$. Since for any X of size at most 3, $|N_G(X)| \geq |X|$, we have that $|Y| \geq |X|$ implying that such a local improvement is not possible. ◀

As mentioned before, a $(4, 3)$-local search gives a 5 approximation.

▶ **Theorem 4.** *Let* $G = (R, B, E)$ *be a bipartite planar graph on red and blue vertex sets* R *and* B, *such that for every subset* $B' \subseteq B$ *of size at most* 4, $|N_G(B')| \geq |B'|$. *Then,* $|B| \leq 5\,|R|$.

The proof is in Appendix A.

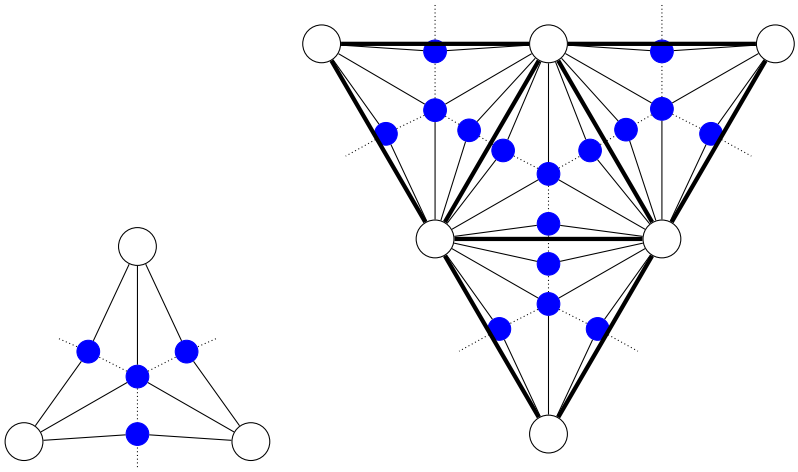

Figure 1 Unshaded vertices correspond to red and shaded to blue vertices. The dotted lines show a triangulation. We tile the triangles in (a) as shown in (b). The ratio of shaded to unshaded vertices goes to 8 as size of the tiling is increased. Connecting the vertices at the boundary of the tiling to a new vertex gives a 4-connected graph.

3 A $\tilde{O}(n^{7/3})$-time Algorithm for $(3, 2)$ Local Search

Our algorithm is based on local search. It starts with a hitting set and repeatedly tries to make local improvements. Let S be a hitting set for $\mathcal{H}(P, \mathcal{D})$. Let $X \subseteq S$ and $Y \subseteq P$. We say that (X, Y) is a *local improvement pair* with respect to S and $\mathcal{H}(P, \mathcal{D})$ if $|Y| < |X|$ and $(S \setminus X) \cup Y$ is a hitting set for $\mathcal{H}(P, \mathcal{D})$. Such a local improvement reduces the size of the hitting set by $|X| - |Y|$. We will refer to this quantity as the *profit* of the local improvement and the local improvement pair. We say that $X \subseteq S$ is *locally improvable* with respect to S and $\mathcal{H}(P, \mathcal{D})$ if there exists a $Y \subseteq P$ such that (X, Y) is a local improvement pair. If (X, Y) is a local improvement pair, we say that Y can *locally replace* X.

Let S be a hitting set for $\mathcal{H}(P, \mathcal{D})$. For any $s \in S$, we denote by $\mathcal{D}(s)$ the set of disks in $\mathcal{D}$ that are hit by s but not by any other point in S. We will call the disks in $\mathcal{D}(s)$ the *personal disks* of s. We will denote the region $\bigcap_{D \in \mathcal{D}(s)} D$ by $R(s)$ and call it the *personal region* of s. The notations $\mathcal{D}(s)$ and $R(s)$ are always with respect to a set system $\mathcal{H}(P, \mathcal{D})$ and a hitting set S. These things that are not explicit in the notation will be clear from the context. We also extend the same definitions for sets of points: for a set $X \subseteq S$, let $\mathcal{D}(X)$ be the set of disks in $\mathcal{D}$ which contain only points of X. We call these the personal disks of X. The personal region of X is $R(X) = \bigcap_{D \in \mathcal{D}(X)} D$. A set of regions $\mathcal{R}$ are *pseudodisks* if they are simply connected and the boundaries of every pair of regions in $\mathcal{R}$ intersect at most twice.

Preparing for the algorithm.

Before presenting our algorithm, we prove a few results useful for describing the algorithm.

▶ **Lemma 5.** *Let S be a hitting set for $\mathcal{H}(P, \mathcal{D})$. If $|S| > 8 \cdot \textsc{Opt}(P, \mathcal{D}) + 3t$, for some integer $t \geq 0$, then there exist $t+1$ disjoint subsets $X_0, \ldots, X_t$ of S, each of which is of size 3 and is locally improvable with respect to S in $\mathcal{H}(P, \mathcal{D})$.*

Proof. The proof is by induction. The statement is true for $t = 0$: if there is no locally improvable set X of size 3, then taking $B = S$ and $R = O$, where O is the optimal hitting set for $\mathcal{H}(P, \mathcal{D})$ and applying Theorem 2, we get that $|S| \leq 8\textsc{Opt}(P, \mathcal{D})$. Assume inductively that the lemma is true for $t - 1$, and let the t disjoint sets of S be $X_0, \cdots, X_{t-1}$. It remains to construct the set X_t. Let $Z = \bigcup_{i=0}^{t-1} X_i$. Let $P' = P \setminus Z$, $\mathcal{D}' = \{D \in \mathcal{D} : D \cap Z = \emptyset\}$ and $S' = S \setminus Z$. Clearly S' is a hitting set for $\mathcal{H}(P', \mathcal{D}')$. Moreover,

▶ Claim 3.1. $\textsc{Opt}(P', \mathcal{D}') \leq \textsc{Opt}(P, \mathcal{D})$.

Proof. Take any hitting set A for $\mathcal{H}((P, \mathcal{D}))$. Then any point $a \in A$ that hits a disk in $\mathcal{D}'$ must belong to P': otherwise $a \in P \setminus P' = Z$, and we had constructed $\mathcal{D}'$ by removing all the disks hit by Z from $\mathcal{D}$. Therefore all the points in A hitting $\mathcal{D}'$ belong to P', and form a hitting set in P' for $\mathcal{H}((P', \mathcal{D}'))$ of size at most $|A|$. ◀

Therefore, $|S'| = |S| - 3t > 8 \cdot \textsc{Opt}(P, \mathcal{D}) \geq 8 \cdot \textsc{Opt}(P', \mathcal{D}')$, and any hitting set for $\mathcal{H}(P, \mathcal{D})$ contains a hitting set for $\mathcal{H}(P', \mathcal{D}')$. Now, using the Theorem 2 for $t = 0$ on S' and $(P', \mathcal{D}')$, the fact that $|S'| > 8 \cdot \textsc{Opt}(P', \mathcal{D}')$ implies that there is set $X_t \subseteq S'$ of size 3 and a set $Y \subseteq P'$ of size 2 such that $(S' \setminus X_t) \cup Y$ is a hitting set for $\mathcal{H}(P', \mathcal{D}')$. This means that $(S' \setminus X_t) \cup Y \cup Z$ is a hitting set for $\mathcal{H}(P, \mathcal{D})$ since all disks in $\mathcal{D} \setminus \mathcal{D}'$ intersect Z. In other words, $(S \setminus X_t) \cup Y$ is a hitting set for $\mathcal{H}(P, \mathcal{D})$ since $S' \cup Z = S$ and $X_t \cap Z = \emptyset$. That is, X_t is locally improvable with respect to S in $\mathcal{H}(P, \mathcal{D})$. Since $X_t \subseteq S'$ and $S' \cap Z = \emptyset$, X_t is disjoint from the other X_i's. ◀

The following key structural property is crucial (proof in Appendix):

▶ **Lemma 6.** *Let S be a hitting set for $\mathcal{H}(P, \mathcal{D})$. Then the personal regions of the points in S form a collection of pseudodisks.*

▶ **Lemma 7.** *Let S be a hitting set for $\mathcal{H}(P, \mathcal{D})$. Suppose that we are given two sets $X \subseteq S$ and $Y \subseteq P$ such that $|Y| = O(1)$, $|X| > 4|Y|$ and for each $x \in X$, Y hits $\mathcal{D}(x)$, the personal disks of x. Then there exists a set $X' \subseteq X$ of size $\Omega(|X|)$ such that (X', Y) is a local improvement pair with respect to S and $\mathcal{H}(P, \mathcal{D})$. Furthermore, given X and Y, X' can be computed in time $O(|X| \log |X|)$.*

Proof. Consider the Delaunay triangulation of the points in X, and let X' be an independent-set in this Delaunay graph. First we show that $(S \setminus X') \cup Y$ is a hitting set for $\mathcal{H}(P, \mathcal{D})$. Consider a disk D that is not hit by $S \setminus X'$. Since D is hit by S (S being a hitting set for $\mathcal{H}(P, \mathcal{D})$), D contains at least one point of X'. If D contains exactly one point $x' \in X'$ then D is hit by Y since $D \in \mathcal{D}(x')$ and Y hits $\mathcal{D}(x')$. Otherwise, D contains at least two points of X', in which case it must contain some point of $x \in X \setminus X' \subseteq S \setminus X'$ since X' is an independent set in the Delaunay triangulation of X (and by the fact that subgraph of the Delaunay graph induced by the set of points of X inside any disk is connected).

The Delaunay triangulation can be constructed in $O(|X| \log |X|)$ time. If $|X| \leq 5|Y|$, i.e. $|X| = O(1)$, we find an independent set of size at least $\lceil |X|/4 \rceil > |Y|$ in the Delaunay graph in $O(1)$ time by brute force; the existence of such an independent set follows from the 4-color theorem on planar graphs. If $|X| > 5|Y|$, we compute a 5-coloring of the Delaunay graph in $O(|X|)$ time and take the largest color class as X'. Thus $|X'| \geq \lceil |X|/5 \rceil > |Y|$.

Therefore $|X'| > |Y|$, and so (X', Y) is a local improvement pair. ◀

The next two lemmas show that one can efficiently preprocess disks to answer containment queries in logarithmic time. Due to the shortage of space, the proof is in the Appendix.

▶ **Lemma 8.** *Let $\mathcal{D}$ be a set of m disks in the plane having a common intersection region, say R. Then the boundary of R is composed of $O(m)$ circular arcs, and can be computed in $O(m \log m)$ expected time. We can also construct, in $O(m \log m)$ time, a data structure which, for any given query point q, answers whether $q \in R$ in $O(\log m)$ time.*

▶ **Lemma 9.** *Let P be a set of n points in the plane and let $\mathcal{D}$ be a set of pseudodisks, the boundary of each being composed of circular arcs. For any constant C, we can compute, for each $p \in P$ that lies in at most C pseudodisks, the exact set of pseudodisks it hits in $O(n \log m)$ time, where m is the total number of arcs in all the pseudodisks.*

The Algorithm.

We now describe our algorithm for computing a small hitting set for $\mathcal{H}(P, \mathcal{D})$. We first compute a hitting set S of size $O(\text{OPT})$. This can be done using the near linear time algorithm of Agarwal-Pan [5]. We also assume that we know the value of $\text{OPT} = \text{OPT}(P, D)$ although it suffices to guess the value of OPT within a $(1+\epsilon)$ factor which can be done in $O(1/\log(1+\epsilon))$ guesses since we know OPT within a constant factor. Throughout this section, we will use n as the total input size. We therefore upper bound $|P|$ and $|D|$ by n.

Next, for a given $\epsilon > 0$, we prune the input so that no point is contained in more than $\Delta = n/(\epsilon \cdot \text{OPT})$ disks. This can be done by iterating over each point $p \in P$ and computing the number of disks $\mathcal{D}' \subseteq \mathcal{D}$ that contain p. If $|\mathcal{D}'| \geq \Delta$, remove the disks in $\mathcal{D}'$ from $\mathcal{D}$ and add the point p to the set Q (which is initially empty). Note that as we go over the points the set $\mathcal{D}$ changes but we do not change the value of Δ. Since each time we add a point to Q, we remove at least Δ disks from $\mathcal{D}$, $|Q| \leq n/\Delta = \epsilon \cdot \text{OPT}$. We can thus add the set Q to our hitting set at the cost of an added ϵ in our approximation factor. This preprocessing procedure takes $O(n^2)$ time (this will not be the bottleneck of our algorithm).

After preprocessing, we pass P and $\mathcal{D}$ to Algorithm 1 which we describe now. It requires an initial hitting set S of size $O(\text{OPT})$ which we obtain from [5]. The goal of Algorithm 1 is to compute a hitting set whose size is at most $(8+\epsilon) \cdot \text{OPT}$. We compute a value $t = |S| - 8 \cdot \text{OPT}$ which indicates how far we are from the solution we seek. As we will see, when t is large, progress can be made quickly. However as we approach the quantity $8 \cdot \text{OPT}$, progress becomes slower and slower. The algorithm uses only local improvements of the type (X, Y) where $|Y| \leq 2$. Throughout the algorithm we maintain for each $D \in \mathcal{D}$, the number of points, N_D, it contains from S. Initially computing N_D for each disk takes $O(n^2)$ time. After that we need to update these quantities only when a local improvement (X, Y) happens. We update N_D as follows: $N_D = N_D - |D \cap X| + |D \cap Y|$. Since $|Y|$ is always at most 2 in our algorithm, naively this takes time $O(n|X|)$ for updating all disks in $\mathcal{D}$. Since such a local improvement decreases the size of the hitting set S by $|X| - 2 = \Omega(|X|)$, the overhead for maintaining N_D is $O(n)$ per improvement. Let `LocallyImprove(`X, Y`)` be the procedure that updates S to $(S \setminus X) \cup Y$ and updates N_D for each disk as mentioned above.

In each iteration of the `while` loop in Algorithm 1, we first construct a range reporting data structure [2] for the points in S so that given any disk D, we can find the set of points in $D \cap S$ in time $O(\log n + |D \cap S|)$. We then use this data structure to compute the personal disks of each $s \in S$ as follows. Iterate over each disk $D \in \mathcal{D}$ and if $N_D = 1$, use the reporting data structure to find the single point $s \in S$ that is contained by D. We then add D to the (initially empty) list of personal disks of s. Since each query takes $O(\log n)$ time, the total time taken to compute the personal disks is $O(n \log n)$. If we find some point $s \in S$ for which $\mathcal{D}(s) = \emptyset$, we can just remove s from the current hitting set. In other words we do a local improvement $(\{s\}, \emptyset)$.

The algorithm iterates over the points in P in random order, considering the possibility of replacing

Algorithm 1: Algorithm for $(8+\epsilon)$-approximation.

Data: A point set P, a set of disks $\mathcal{D}$, a hitting set S of $\mathcal{H}(P,\mathcal{D})$ with $|S| = O(\text{OPT}(\mathcal{H}(P,\mathcal{D})))$, the size of the optimal hitting set $\text{OPT} = \text{OPT}(\mathcal{H}(P,\mathcal{D}))$, and a parameter $\epsilon > 0$.

1 For each disk $D \in \mathcal{D}$ compute $N_D = |D \cap S|$ // takes $O(n^2)$ time
2 **while** $t = |S| - 8 \cdot \text{OPT} > \epsilon \cdot \text{OPT}$ **do**
3 Construct a range reporting data structure for S for disk ranges
4 For each $s \in S$ compute $\mathcal{D}(s) = \{D \in S : D \cap s = \{s\}\}$// use range reporting
5 **if** $\mathcal{D}(s) = \emptyset$ *for some* $s \in S$ **then**
6 LocallyImprove($\{s\}, \emptyset$) // s is dropped from the hitting set
7 **continue** // with the next iteration of the while loop on line 2.
8 π = A random permutation of the points in P
9 **for** i = *1 to* $|P|$ **do**
10 $p_1 = \pi_i$
11 **for** *each* $s \in S$ **do**
12 Compute: $\mathcal{D}'(s) = \{D \in \mathcal{D}(s) : p_1 \notin D\}$, $R'(s) = \bigcap_{D \in \mathcal{D}'(s)} D$
 // The above loop takes $O(n \log n)$ time
13 Let $\mathcal{R}' = \{R'(s) : s \in S\}$ // $\mathcal{R}'$ is a set of pseudodisks
14 $M = \{s \in S : R'(s) = \emptyset\}$
15 **for** *each* $p \in P$ **do**
16 Compute $\alpha(p)$ s.t. $0.9 \cdot \text{depth}(p, \mathcal{R}') \le \alpha(p) \le \text{depth}(p, \mathcal{R}')$
 // depth(p,$\mathcal{R}'$) denotes the number of regions in $\mathcal{R}'$ containing p
17 Let $q = \arg\max_{p \in P} \alpha(p)$
18 Set $\beta = \max\{\sqrt{t}, \epsilon t \cdot \text{OPT}/n\}$
19 **if** $|M|/5 + \alpha(q) \ge \frac{i\beta}{Cn \log n}$ **then**
20 Compute $S'(q) = \{s \in S : q \in R'(s)\}$ // Note that $|S'(q)| = \text{depth}(q, \mathcal{R}')$
21 **if** $|S'(q) \cup M| \ge 9$ **then**
22 Compute an independent set $X \subseteq S'(q) \cup M$ in the Delaunay triangulation of $S'(q)$ of size at least 3 and $\Omega(|S'(q) \cup M|)$ // $O(n \log n)$ time
23 LocallyImprove($X, \{p_1, p_2 = q\}$)
24 **break** // exit for loop
25 **else**
26 For each $p_2 \in P$, set $S'(p_2) = \{s \in S : p_2 \in R'(s)\}$ // $O(n \log n)$ time
 // Since $|S'(q) \cup M| \le 8$, $|S'(p_2) \cup M| \le \lfloor 8/0.9 \rfloor = 8$ for all $p_2 \in P$
27 Enumerate all pairs (X, p_2) where $p_2 \in P$, $X \subseteq S'(p_2) \cup M$ and $|X| \le 3$
28 **if** *for any* (X, p_2) *enumerated,* $(X, \{p_1, p_2\})$ *is a local improvement pair* **then**
29 LocallyImprove($X, \{p_1, p_2\}$))
30 **break** // exit for loop

each point in a local-improvement step. Say the current point being considered is p_1; the goal is to find a point p_2 so that $\{p_1, p_2\}$ can replace a large set $X \subseteq S$, i.e., a local improvement pair $(X, \{p_1, p_2\})$ of large profit. If we can find such a profitable local improvement, we make the improvement, exit from the `for` loop, and continue with the next iteration of the `while` loop. Otherwise, we continue with the next point in the random ordering. For any pair of points $Y = \{p, q\} \subseteq P$, denote by $\rho(Y)$ the number of points in S all of whose personal disks are hit by Y. For a point $p \in P$, we use $\rho(p)$ to denote $\max_{q \in P \setminus S} \rho(\{p, q\})$. Call a point $p \in P$ *useful* if there exists some $q \in P$ so that for some $X \subseteq S$, $(X, \{p, q\})$ is a local improvement pair.

Analysis of the Algorithm.

▶ **Lemma 10.** *If p_1 is useful, we can compute in $O(n \log^2 n)$ time a local improvement of profit $\Omega(\rho(p_1))$.*

Proof. Let us start by considering how we could compute $\rho(p_1)$. In order to compute $\rho(p_1)$, we need to find a point q so that the number of points $s \in S$ whose personal disks are hit by $\{p_1, q\}$ is maximized. To do this, we first compute for each $s \in S$, the set $\mathcal{D}'(s)$ of disks in $\mathcal{D}(s)$ that are not hit by p_1. For each $s \in S$, we then construct the region $R'(s)$ by taking the intersection of the disks in $\mathcal{D}'(s)$. Let $\mathcal{R}' = \{R'(s) : s \in S\}$. For some $s \in S$, $\mathcal{D}'(s)$ may be empty and consequently some of the regions in $\mathcal{R}'$ are empty. Let $M = \{s \in S : \mathcal{D}'(s) = \emptyset\}$. The personal disks of the points in M are hit by p_1 alone. The regions in $\mathcal{R}'$ define an arrangement of pseudodisks (Lemma 6). In this arrangement we seek to find a point $q \in P$ of maximum depth. However, instead of finding a point with maximum depth, we find a point whose depth is within a constant factor of the maximum. We construct, in $O(n \log n)$ time, an approximate depth query data structure for the pseudodisks in $\mathcal{R}'$ using Corollary 5.9 in [7] with a constant $\epsilon' \leq 0.1$. This takes $O(n \log n)$ time. Then, for each point $p \in P$, we compute a value $\alpha(p)$ s.t. $0.9 \text{ depth}(p, \mathcal{R}') \leq \alpha(p) \leq \text{depth}(p, \mathcal{R}')$ where $\text{depth}(p, \mathcal{R}')$ denotes the depth of p in the arrangement of regions in $\mathcal{R}'$. This takes $O(\log^2 n)$ time per point and so the overall time taken is $O(n \log^2 n)$. We then take the point p with the maximum $\alpha(p)$ as q. Observe that $|M| + \alpha(q) = \Theta(\rho(p_1))$.

We first compute the set $S'(q) = \{s \in S : q \in R'(s)\}$. Note that $|S'(q)| = \text{depth}(q, \mathcal{R}') \geq \alpha(q)$. There are two cases to consider:

Case 1: $|S'(q) \cup M| > 8$. In this case, we set $p_2 = q$ and let $Y = \{p_1, p_2\}$. Using Lemma 7, we can find a subset $X \subseteq S'(q) \cup M$ so that $X = \Omega(|S'(q) \cup M|)$ so that $(X, \{p_1, p_2\})$ is a local improvement pair. Note that $|X|$ is $\Omega(\rho(p_1))$. Thus in this case, we conclude that p_1 is useful and indeed we have found a local improvement that decreases the size of the current hitting set by $\Omega(\rho(p_1))$.

Case 2: $|S'(q) \cup M| \leq 8$. In this case $S'(p) \leq \lfloor 8/0.9 \rfloor = 8$ for all $p \in P$. This means that $\rho(p_1) = O(1)$ and we just need to find one set X of size 3 and a point p_2 so that $(X, \{p_1, p_2\})$ is a local improvement pair. Using Lemma 9, we compute the set $S'(p)$ for all $p \in P$ in $O(n \log n)$ expected time. For each $p_2 \in P$, we need to check if there is any subset X in $S'(p_2) \cup M$ of size 3 so that $(X, \{p_1, p_2\})$, is a local improvement pair. Since $|S'(p_2) \cup M| \leq 8$, there are at most $\binom{8}{3}$ subsets $X \subseteq S'(p_2) \cup M$ for which we need to check if $(X, \{p_1, p_2\})$ is a local improvement pair. Thus there are $O(n)$ pairs of the form $(X, \{p_1, p_2\})$, where $|X| = 3$, that we need to check. For a particular pair of this form, we basically need to verify that all the disks in $\mathcal{D}$ whose intersection with S is a subset of X are hit by either p_1 or p_2. To make things simpler, we first remove from $\mathcal{D}$ all the disks that are hit by p_1 and obtain a set $\mathcal{D}' \subseteq \mathcal{D}$. Now, we need to verify for all disks in $\mathcal{D}$ whose intersection with X is a subset of X that they are hit by p_2. All the $O(n)$ pairs can be checked in $O(n \log n)$ time as follows.

We construct a data structure that will help us do the checking for all the $O(n)$ pairs of the form $(X, \{p_1, p_2\})$. We have already constructed a range reporting data structure on S for disk ranges. Additionally, use a dictionary data structure (based on balanced binary trees) in which the keys are subsets of S of size at most 3 and the value corresponding to a key U is a list of disks $D \in \mathcal{D}'$ s.t. $D \cap S = U$. We start with an empty dictionary. We then go over each disk $D \in \mathcal{D}'$ one by one and if $N_D \leq 3$, we use the range reporting data structure to get $U = D \cap S$ in $O(\log n)$ time. We search the dictionary for U and if it is found, we add D to its list. If no entry is found, we create an entry for U with a single element d in its list. Note that since the number of (≤ 3)-sets that can be obtained from set of n points by intersecting it with a set of disks is linear in the number of points [20], the number of distinct keys in the dictionary is $O(n)$. We go over each key U and construct the region $R'(U)$ by taking the intersection of all the disks in the list associated with U. Note that $R'(U)$ can be constructed in $O(m \log m)$ time where m is the size of the list associated with U using Lemma 8. Since each disk is in the list of at most one U, the overall time is $O(n \log n)$. In the same amount of time, for each key U, we set up a data structure that allows us to check if a query point q is in $R'(U)$ using Lemma 8. Now, to check if a pair $(X, \{p_1, p_2\})$ is an improvement pair, we go over all subsets $U \subseteq X$ and check if $p_2 \in R'(U)$. The time spent for any pair is now $O(\log n)$. Therefore checking all the $O(n)$ pairs takes $O(n \log n)$ time.

If we find that none of the pairs we checked are local improvement pairs, then we can conclude that p_1 is not useful. ◀

The following lemma shows to find a profitable improvement. Let $\beta = \max\{\sqrt{t}, \epsilon t \cdot \textsc{Opt}/n\}$.

▶ **Lemma 11.** *There exists a $k > 0$ such that there are at least $\Omega(\beta/k)$ useful points $p \in P$ with $\rho(p) \geq k$.*

Proof. By Lemma 5, there exists $\Omega(t)$ local improvement pairs $(X_0, Y_0), \ldots$ where the X_i's are disjoint subsets of S but the Y_i's need not be disjoint. Each X_i is of size 3 and each Y_i is of size 2. For any pair of points $Y = \{p_1, p_2\} \subseteq P$, if (X_i, Y) is a local improvement pair among the $\Omega(t)$ pairs, then we say that X_i is a triple assigned to the pair Y. Define the weight of Y as the number of triples assigned to it and denote it by $W(Y)$. The total weight of all pairs is then $\Omega(t)$.

Call a pair Y to be of type i if $2^{i-1} \leq W(Y) < 2^i$, for $i = 1, \ldots, O(\log t)$. If $W(Y) = 0$ then we say that Y is of type 0. Since the total weight of all pairs is $\Omega(t)$, there must be some $j > 0$ so that the total weight of the pairs of type j is $\Omega(t/2^j)$. Let $Q = \bigcup_Y \{Y \mid Y \text{ is of type } j\}$.

There are two lower bounds on the size of Q. First, since the total weight of the pairs of type j is $\Omega(t/2^j)$, and each pair has weight less than 2^j, the number of pairs is $\Omega(t/2^{2j})$, and hence $|Q| = \Omega(\sqrt{t}/2^j)$. On the other hand, for any local improvement pair (X_i, Y) where Y is of type j, take any point $x \in X_i$. Since $\mathcal{D}(x)$ is non-empty, any disk $D \in \mathcal{D}(x)$ contains at least one point in Y. Therefore any such local improvement pair leads to an incidence between a point in Q and a disk in $\mathcal{D}$. Note that since the X_i's are disjoint these are distinct incidences. Thus there are $\Omega(t/2^j)$ incidences. Since by assumption no point in P, and therefore no point in Q, is in more than $n/(\epsilon \cdot \textsc{Opt})$ disks in $\mathcal{D}$, we have that $|Q| = \Omega(\epsilon t \cdot \textsc{Opt}/2^j n)$.

Therefore, $|Q| = \Omega\left(\max\{\epsilon t \cdot \textsc{Opt}/2^j n, \sqrt{t}/2^j\}\right) = \Omega(\beta/2^j)$. Observe that each $p \in Q$ is useful and $\rho(p) \geq 3 \cdot 2^j$. The lemma is therefore true for $k = 2^j$. ◀

We can now analyze the running time of our algorithm.

Running time: Preprocessing takes $O(n^2)$ time but this is dominated by the running time of Algorithm 1. Consider a single iteration of the `while` loop in Algorithm 1. If we find some point

$s \in S$ for which $\mathcal{D}(s) = \emptyset$, we drop s from the current hitting set. This way we have improved the size of the hitting set at the cost of $O(n \log n)$ time. The total time spent on such improvements is at most $O(\text{OPT}\ n \log n) = O(n^2 \log n)$.

Otherwise, call a single iteration of the `while` loop *lucky* if the following is true:

$$\exists i \text{ such that the point } \pi_i \text{ is useful and } \frac{i}{\rho(\pi_i)} \leq \frac{Cn}{\beta}$$

for some constant C.

▶ Claim 3.2. Probability that any iteration of the `while` loop is lucky is at least $1/2$.

Proof. By Lemma 11, there exists a k such that there are $\Omega(\beta/k)$ points, say the set U, with $\rho(p) \geq k$. Consider the smallest index i s.t. $\pi_i \in U$. The expected value of i is $O(nk/\beta)$. Therefore, with probability at least $\frac{1}{2}$, $i \leq Cnk/\beta$ for some large enough C. Then,

$$\frac{i}{\rho(\pi_i)} \leq \frac{Ckn/\beta}{k} = \frac{Cn}{\beta}$$

◀

▶ Claim 3.3. For a lucky iteration of the `while` loop, let λ be the reduction in size of the current hitting set, and σ the time spent in this iteration. Then $\sigma/\lambda \leq Cn^2 \log^2 n/\beta$.

Proof. As we go over the points in random over, for the current point $\nu = \pi_i$, we estimate $\rho(\nu)$ which allows us to check if $i/\rho(\nu) \leq Cn/\beta$. If so, assuming that the point ν is useful, we decrease the size of the current hitting set by $\Omega(\rho(\nu))$. If $i/\rho(\nu) > Cn/\beta$ or we discover that ν is not useful we move to the next point in the random order. However, since the iteration of the `while` loop is lucky, we will find some point $\nu = \pi_i$ which is useful and for which $i/\rho(\nu) \leq Cn/\beta$. For this point ν, we find a local improvement involving ν of value $\Omega(\rho(\nu))$ and the current iteration of the `while` loop ends. The total time spent in this iteration is $\sigma = O(i \cdot n \log^2 n)$ since we have seen i points so far and for each point we spend $O(n \log^2 n)$ time. The reduction in the size of the current hitting set is $\lambda = \Omega(\rho(\nu))$. Thus $\sigma/\lambda \leq \frac{i}{\rho(\nu)} \cdot n \log^2 n \leq Cn^2 \log^2 n/\beta$. ◀

Since any iteration of the `while` loop is lucky with probaility at least 0.5 and we can assume that all the iterations are lucky. This does not change the running time by more than a factor of 2.

▶ Claim 3.4. The expected time taken to halve t is $O(n^{7/3} \log^2 n\epsilon^{-1/3})$.

Proof. Claim 3.3 tells us that the amortized amount of time spent for the reducing the size of the current hitting set by 1 is $O(n^2 \log^2 n/\beta)$. Since β is an increasing function of t, this decreases with t. However, t does not change by more than a factor of 2 until it is halved. So, the expected time for t to be halved is $O(t/2 \cdot n^2 \log^2 n/\beta)$. Now, $t/\beta = \min\{\sqrt{t}, n/(\epsilon \cdot \text{OPT})\}$. Since $t = O(\text{OPT})$, $t/\beta = O(min\{\sqrt{\text{OPT}}, n/(\epsilon \cdot \text{OPT})\} = O((n/\epsilon)^{1/3})$. Thus the expected time to halve t is $O(n^{7/3} \log^2 n\epsilon^{-1/3})$. ◀

Since the initial value of t is $O(\text{OPT})$, there are $O(\log 1/\epsilon)$ halving rounds until $t \leq \epsilon \cdot \text{OPT}$. Thus, the expected running time of the Algorithm 1 is $O(n^{7/3} \log^2 n\ \epsilon^{-1/3} \log(1/\epsilon))$. Finally, since we need to run Algorithm 1 for $O(1/\log(1+\epsilon))$ guesses for OPT, the overall running time is $O(n^{7/3} \log^2 n\ \epsilon^{-1/3} \log(1/\epsilon)/\log(1+\epsilon))$. For a fixed small value of ϵ, this is $O(n^{7/3} \log^2 n)$.

References

[1] Rashmisnata Acharyya, Manjanna Basappa, and Gautam K. Das. Unit disk cover problem in 2d. In *Computational Science and Its Applications - ICCSA 2013 - 13th International Conference, Ho Chi Minh City, Vietnam, June 24-27, 2013, Proceedings, Part II*, pages 73–85, 2013.

[2] Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *SODA*, pages 180–186, 2009.

[3] Pankaj K. Agarwal, Esther Ezra, and Micha Sharir. Near-linear approximation algorithms for geometric hitting sets. *Algorithmica*, 63(1-2):1–25, 2012.

[4] Pankaj K. Agarwal, János Pach, and Micha Sharir. State of the Union (of Geometric Objects): A Review, 2007.

[5] Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Symposium on Computational Geometry*, page 271, 2014.

[6] Christoph Ambühl, Thomas Erlebach, Matús Mihalák, and Marc Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *APPROX-RANDOM*, pages 3–14, 2006.

[7] Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008.

[8] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

[9] Gruia Călinescu, Ion I. Mandoiu, Peng-Jun Wan, and Alexander Zelikovsky. Selecting forwarding neighbors in wireless ad hoc networks. *MONET*, 9(2):101–111, 2004.

[10] Paz Carmi, Matthew J. Katz, and Nissan Lev-Tov. Covering points by unit disks of fixed location. In *ISAAC*, pages 644–655, 2007.

[11] Francisco Claude, Gautam K. Das, Reza Dorrigiv, Stephane Durocher, Robert Fraser, Alejandro López-Ortiz, Bradford G. Nickerson, and Alejandro Salinger. An improved line-separable algorithm for discrete unit disk cover. *Discrete Math., Alg. and Appl.*, 2(1):77–88, 2010.

[12] Michael B. Dillencourt and Warren D. Smith. Graph-theoretical conditions for inscribability and delaunay realizability. *Discrete Mathematics*, 161(1–3):63 – 77, 1996.

[13] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Inf. Process. Lett.*, 95:358–362, 2005.

[14] Eti Ezra, Dan Halperin, and Micha Sharir. Speeding up the incremental construction of the union of geometric objects in practice. *Comput. Geom.*, 27(1):63–85, 2004.

[15] Robert Fraser. *Algorithms for Geometric Covering and Piercing Problems*. PhD thesis, University of Waterloo, 2012.

[16] Shashidhara K. Ganjugunte. *Geometric Hitting Sets and Their Variants*. PhD thesis, Duke University, 2011.

[17] Alejandro López-Ortiz Gautam K. Das, Robert Fraser and Bradford G. Nickerson. On the discrete unit disk cover problem. *International Journal on Computational Geometry and Applications*, 22(5):407–419, 2012.

[18] D. S. Hochbaum and W. Maass. Fast approximation algorithms for a nonconvex covering problem. *J. Algorithms*, 8(3):305–323, 1987.

[19] Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.*, 11(3):555–556, 1982.

[20] J. Matousek. *Lectures in Discrete Geometry*. Springer-Verlag, New York, NY, 2002.

[21] Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[22] Thomas Ottmann, Peter Widmayer, and Derick Wood. A fast algorithm for the boolean masking problem. pages 249–268, 1985.

[23] János Pach and Micha Sharir. Combinatorial Geometry with Algorithmic Applications – the Alcala Lectures, 2006.

[24] E. Pyrga and S. Ray. New existence proofs for epsilon-nets. In *Proceedings of Symposium on Computational Geometry*, pages 199–207, 2008.

14 Rferences

[25] R. Raz and M. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of STOC*, pages 475–484, 1997.

A Proofs

Proof of Lemma 8. Let p be a point in the region R and D be disk in $\mathcal{D}$. Consider the function $f_D(\theta)$, for $\theta \in [0, 2\pi)$, to be the distance from p to the boundary of D in the direction θ (i.e., along a ray emanating from p at an angle θ to the positive x axis). Clearly the graph of $f_D(\theta)$ with θ plotted along the x-axis is an x-monotone curve which we denote by Γ_D. Furthermore, for any two disks $D, D' \in \mathcal{D}$ the curves Γ_D and $\Gamma_{D'}$ intersect at most twice since the boundaries of any two disks intersect at most twice. The curves in $\{\Gamma_D : D \in \mathcal{D}\}$ therefore form a set pseudo-parabolas. The number of arcs in the boundary of R is equal to the size of the lower envelope of the arrangement of curves, which is $O(m)$ due to the linear union complexity of pseudo-parabolas [23, 4].

The boundary of R can be computed by a randomized incremental construction in $O(m \log m)$ in the same way as the union of a set of m regions with linear union complexity is computed in the same amount of time. There is also a deterministic algorithm which takes $O(m \log^2 m)$ time. It basically splits the disks in $\mathcal{D}$ into disjoint sets of half the size, computes the intersection region for each, and then takes the intersection of the two intersection regions - which can be computed using a circular sweep. For details see [22, 14, 4] and the references therein.

Once R has been computed, we can easily set up a data structure that checks for a query point q whether $q \in R$. The region R is almost like a convex polygon except that its sides are circular arcs. We can take one vertex v of the region R and join it with a chord to each of the other vertices. The chords define a linear order. Given a query point q, we can determine, by binary search, two consecutive chords so that the cone defined by v and the rays emanating from v along the two chords contains q. Next we just need to check whether q lies on the same side of the circular arc bounding R in that cone, as v. Setting up this data structure takes $O(m)$ time and the query time is $O(\log m)$ as required.

◀

Proof of Lemma 9. We first compute the shallow levels (of depth at most C) of the arrangement of pseudodisks $\mathcal{D}$. This can be done using a randomized incremental construction in exactly the same way as the union of a set of pseudodisks is computed. This takes $O(m \log m)$ time since the overall complexity of the shallow levels (for constant C) is linear. For details see [22, 14, 4]. Once the shallow levels are computed, we set up a point location data structure. Then in $O(\log m)$ time per point, we can determine for each point exactly which of the at most C pseudodisks it is contained in. ◀

Proof of Theorem 4. Let $n_b = |B|$ and $n_r = |R|$. As in the proof of Theorem 2, we construct a graph G' such that every vertex of B has degree at least 2. As before we can show that $n_{b_{\geq 3}} \leq 2n_r$.

We now show that $n_{b_2} \leq 3n_r$. We construct the graph H as before. Since the vertex set of H is a subset of R, the number of vertices in H is at most n_r. Each edge in H has either one or two blue vertices corresponding to it. We assign to each edge a weight equal to the number of blue vertices corresponding to it. The total weight of all edges in H gives n_{b_2}. Recall that H is planar and therefore we can complete it to a triangulation T by adding additional edges of weight 0 so that the total weight of the edges is unchanged. Consider any (triangular) face t of this triangulation. The total weight of the edges of this face cannot be 4 or more since that corresponds to the existence of four or more blue vertices with a neighborhood of size at most 3 (the vertices of t), violating the conditions of the theorem.

We distribute the weight of the edges of T to the faces of T as follows: each edge distributes its

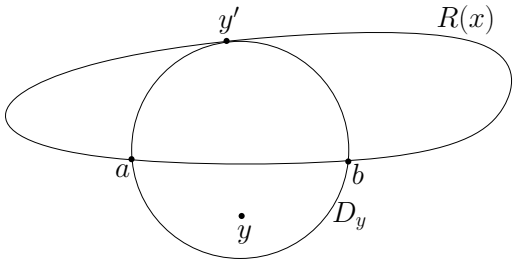

Figure 2 Case 2 in the proof of Lemma 6.

weight equally among the two faces adjacent to it. Since the edges surrounding any face have total weight at most 3, each face gets a weight of at most 1.5. Since the number of faces in the triangulation is at most twice the number of vertices ($2n_r$), we obtain that the total weight of all the faces is at most $3n_r$. In other words, $b_{n_2} \leq 3n_r$. Thus $b = n_{b_{\geq 3}} + n_{b_2} \leq 5n_r$. ◀

Proof of Lemma 6. First observe that since each personal region $R(s)$ is an intersection of disks, it is convex. We show that for any two points $x, y \in S$, $R(x)$ and $R(y)$ are non-piercing i.e., the regions $R(x) \setminus R(y)$ and $R(y) \setminus R(x)$ are connected. Since the boundaries of two convex regions that are non-piercing cannot intersect in more than two points, we conclude that the regions form a collection of pseudodisks.

Assume for contradiction that $R(x)$ and $R(y)$ are piercing and without loss of generality that $R(x) \setminus R(y)$ has at least two connected components. The point x lies in one of these components and let x' be a point in a different component of $R(x) \setminus R(y)$. Since x' does not lie in $R(y)$, there must be a disk $D_y \in \mathcal{D}(y)$ that does not contain x'. Since no disk in $\mathcal{D}(y)$ contains x, D_y also does not contain x. Since D_y contains $R(y)$, this implies that x and x' are in different connected components of $R(x) \setminus D_y$. In other words $R(x) \setminus D_y$ is not connected. There are two cases to consider now:

Case 1: $D_y \setminus R(x)$ **is not connected.** In this case y lies in one of the connected components of $D \setminus R(x)$. Let y' be a point in one of the other components. Since $y' \notin R(x)$, there is some disk $D_x \in \mathcal{D}(x)$ which does not contain y' and by assumption does not contain y. Since D_x contains $R(x)$, this implies that $D_y \setminus D_x$ is not connected, a contradiction as D_x, D_y are disks.

Case 2: $D_y \setminus R(x)$ **is connected.** In this case, since $R(x) \setminus D_y$ is not connected, the situation is as shown in Figure 2. There is at least one point y' where the boundaries of D_y and $R(x)$ intersect but do not cross. Furthermore, y' does not lie on the boundary of $D_y \setminus R_x$. Since y' lies on the boundary of R_x, it lies on the boundary of some disk $D_x \in \mathcal{D}(x)$. Note that D_x contains R_x but does not contain y. Therefore, D_x must intersect the boundary of D_y at least twice in the portion of ∂D_y that lies outside R_x, i.e., the arc between a and b (shown in the figure) in counterclockwise direction. The boundaries of D_x and D_y then intersect at least three times, a contradiction for disks. ◀