Near-Linear and Parameterized Approximations for Maximum Cliques in Disk Graphs

Jie Gao
Computer Science Department, Rutgers University

Paweł Gawrychowski
Institute of Computer Science, University of Wrocław

Panos Giannopoulos
Department of Computer Science, City St George's, University of London

Wolfgang Mulzer
Institut für Informatik, Freie Universität Berlin

Satyam Singh
Department of Computer Science, Aalto University

Frank Staals
Department of Information and Computing Sciences, Utrecht University

Meirav Zehavi
Institute for the Theory of Computing, Ben Gurion University of the Negev

Abstract

A *disk graph* is the intersection graph of (closed) disks in the plane. We consider the classic problem of finding a maximum clique in a disk graph. For general disk graphs, the complexity of this problem is still open, but for unit disk graphs, it is well known to be in P. The currently fastest algorithm runs in time $O(n^{7/3+o(1)})$, where n denotes the number of disks [19, 28]. Moreover, for the case of disk graphs with t distinct radii, the problem has also recently been shown to be in XP. More specifically, it is solvable in time $O^*(n^{2t})$ [28]. In this paper, we present algorithms with improved running times by allowing for approximate solutions and by using randomization:

(i) for unit disk graphs, we give an algorithm that, with constant success probability, computes a $(1-\varepsilon)$-approximate maximum clique in expected time $\tilde{O}(n/\varepsilon^2)$; and

(ii) for disk graphs with t distinct radii, we give a parameterized approximation scheme that, with a constant success probability, computes a $(1-\varepsilon)$-approximate maximum clique in expected time $\tilde{O}(f(t)\cdot(1/\varepsilon)^{O(t)}\cdot n)$, for some (exponential) function $f(t)$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Maximum Clique, Disk Graphs, Unit Disk Graphs, FPT Approximation

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.23

Funding *Jie Gao*: Research on this paper was supported by NSF under grants CNS-2515159, IIS-2229876, DMS-2220271, DMS-2311064, and CCF-2118953.
Paweł Gawrychowski: Research on this paper was supported by the Polish National Science Centre under grant 2023/51/B/ST6/01505.
Satyam Singh: Research on this paper was supported by the Research Council of Finland, Grant 363444.
Meirav Zehavi: Research on this paper was supported by he Israel Science Foundation under grant 1470/24, and European Research Council under grant 101039913 (PARAPATH).

1 Introduction

A *disk graph* is the intersection graph of closed disks in the plane, where the vertices are the disks, and two vertices are connected by an edge if and only if the two corresponding disks intersect. A *unit disk graph* is a disk graph in which all disks have the same radius, which can be assumed to be 1. Disk graphs and unit disk graphs are probably the most extensively studied classes of geometric intersection graphs [5, 6, 8, 15, 22, 35]. In particular, they are popular models of wireless communication networks, where each disk represents a wireless station, and the respective radii encode the transmission ranges. Theoretically, unit disk graphs and disk graphs are of special interest, because they generalize the familiar class of planar graphs. Indeed, the well-known circle-packing theorem (also called the the Koebe–Andreev–Thurston theorem [30]) shows that every planar graph is a disk graph. Unlike planar graphs, disk graphs (or unit disk graphs) can be dense, with potentially $\Theta(n^2)$ edges. Thus, algorithms on disk graphs often use an implicit representation (of the coordinates and radii of the disks) and derive edges on-demand.

Due to the special structure of disk graphs, it is an active research direction to study efficient approximation algorithms for classical NP-hard problems on disk graphs or unit disk graphs, such as maximum independent set [12, 25] and minimum dominating set [23, 32]. In this paper, we study the maximum clique problem on unit disk graphs and disk graphs. For a given graph $G = (V, E)$, the problem asks for a vertex set $S \subseteq V$ of maximum cardinality such that the induced subgraph on S is a clique. The maximum clique problem on general graphs is a classical NP-hard problem and is also hard to approximate [20]. However, maximum clique can be solved in polynomial time on unit disk graphs and the problem has been actively studied recently. We first review prior work on this problem and then present our new results.

Related work. Clark, Colbourn, and Johnson [15] gave an elegant $O(n^{4.5})$-time algorithm for finding a maximum clique in a unit disk graph. The algorithm guesses in quadratic time the pair of most distant (diameter) disk centers in a maximum clique and considers the subgraph induced by all the disks whose centers are at most as distant from both candidate diameter centers. The induced graph is co-bipartite, hence computing a maximum clique is equivalent to finding a maximum independent set in the complement bipartite graph. This in turn reduces to finding a maximum matching. By reducing the necessary number of pairs of centers to search for and/or reducing the running time for finding a maximum independent set, the running time has been successively improved to $O(n^{3.5} \log n)$ by Breu [7], $O(n^3 \log n)$ by Eppstein [18], $O(n^{2.5} \log n)$ by Espenant, Keil and Mondal [19] and, most recently, to $O(n^{7/3+o(1)})$ by Chan [11] as noted in Keil and Mondal [28]. In particular, the latter result is based on the clever divide and conquer approach by Espenant, Keil and Mondal [19], which considers only a linear number of center pairs, and on the observation that, for each such pair, one can use a small bi-clique cover of size[1] $\tilde{O}(n^{4/3})$ of the resulting bipartite graph to compute a maximum matching in time almost linear to the size of cover. Very recently, Tkachenko and Wang [35] gave an algorithm that runs in time $O(n \log n + nK^{4/3+o(1)})$, where K is the size of the maximum clique; this is an improvement to the above bound by Chan [11] when $K = o(n)$.

For general disk graphs, the complexity of computing the maximum clique has been a long-standing open problem [3, 19, 22]. Recently, Keil and Mondal [28] considered the case

[1] The standard $\tilde{O}(\cdot)$ notation hides polylog factors in n.

where there are t distinct disk radii and gave an algorithm that runs in polynomial time for every constant t. In particular, the running time is $O(2^t n^{2t}(f(n)+n^2))$, where $f(n)$ is the time to compute a maximum matching in an n-vertex bipartite graph. A maximum matching in a bipartite graph can be computed in time $\tilde{O}(m)$, where m is the number of edges in the graph, using the near-linear time max-flow algorithm in the recent breakthrough in [13] with standard reductions to bipartite matching, e.g., the $O(m \log m)$-time fractional flow rounding technique in [26]; alternatively, it can be computed in time $\tilde{O}(n^2)$ by a combinatorial algorithm [14]. Further, the problem admits a randomized EPTAS[2] with a running time of $2^{\tilde{O}(1/\varepsilon^3)} n^{O(1)}$ and an exact sub-exponential time algorithm [5].

Finally, the problem has also been studied for intersection graphs of other classes of objects in the plane and has been shown to be NP-hard for rays [9], strings [29], ellipses and triangles [3], and for sets of both rectangles and unit disks [6].

Our contribution. The above recent exciting results and the absence of any non-trivial lower bounds for the problem leave some natural open questions.

For improving the currently best $O(n^{7/3+o(1)})$-time bound for unit disk graphs, any algorithm that follows the classical line of approach by Clark, Colbourn, and Johnson [15] would effectively need to reduce the number of candidate disk centers to $o(n)$ or compute a maximum matching in the resulting complement bipartite graph in $o(n^{4/3})$ time. Both of these options seem challenging. On the other hand, the algorithm by Tkachenko and Wang [35] does not bring an improvement for instances with $\Omega(n)$-size cliques, and dense instances are in fact the hardest for the problem, see for example [5].

In this paper, we show that further improvement is possible, albeit by allowing for $(1-\varepsilon)$-approximate solutions and by using randomization. Our first result is as follows:

(i) For unit disk graphs, there is an algorithm that computes with constant success probability a $(1-\varepsilon)$-approximate maximum clique in expected time $\tilde{O}(n/\varepsilon^2)$; this result is presented in Section 3.

In particular, we first random sample two disk centers inside a constant-size neighborhood of a properly defined grid-cell. We show that these centers can help us identify a small region that contains (the centers of) a $(1-\varepsilon)$-approximate maximum clique of the neighborhood, with $\Omega(\varepsilon)$ probability. Moreover, the disks with centers in that small region induce a co-bipartite graph. To save time, our second step, which is given in Section 2, is to actually compute a $(1-\varepsilon)$-approximate maximum clique in this co-bipartite graph in time that is almost linear in the number of disks of the graph.

Our second result is for the general disk graph setting. As mentioned above, the EPTAS in [5] runs in $2^{\tilde{O}(1/\varepsilon^3)} \cdot O(n^4)$ time while, for disk graphs with t distinct radii, the algorithm by Keil and Mondal [28] runs in $O^*(n^{2t})$ time[3]. This raises the question of whether such exponential dependencies w.r.t. t (on n) and ε can be improved or even removed altogether. We show that this is possible by considering parameterized $(1-\varepsilon)$-approximations w.r.t. t and ε and, as in the case of unit disk graphs, by using randomization.

(ii) For disk graphs with t distinct radii, there is an efficient parameterized approximation scheme (EPAS)[4] that computes with a constant success probability a $(1-\varepsilon)$-approximate

[2] EPTAS stands for efficient polynomial-time approximation scheme.
[3] The standard notation $O^*(\cdot)$ hides polynomial factors.
[4] An efficient parameterized approximation scheme (EPAS), for some parameter k, is a $(1-\varepsilon)$-approximation algorithm that runs in $f(k,\varepsilon)n^{O(1)}$ time [21].

maximum clique in $\tilde{O}(f(t) \cdot (1/\varepsilon)^{O(t)} \cdot n)$ expected time, for some (exponential) function $f(t)$; this result is presented in Section 4.

Our algorithm is based on the main idea behind the exact algorithm by Keil and Mondal. Suppose that for each of the radii one can guess the leftmost and rightmost disks in an optimal solution. Then, one can find two sets of disks L and R such that $L \cup R$ induces a co-bipartite graph and contains a maximum clique. The algorithm by Keil and Mondal [28] enumerates all possible choices to find the leftmost and rightmost disks for each radius. To speed up the running time for $(1-\varepsilon)$-approximation, we bring in three more ideas. First, we can assume w.l.o.g that for disks of radius r_i that appear in the optimal clique, at least $\Theta(\varepsilon k^*/t)$ of them appear in the max clique, where k^* is the optimal clique size, which can be approximated by a factor of $1/5$ in $O(n \log n)$ time (see Section 4.2) – otherwise, we can skip disks of radius r_i completely without losing more than ε fraction. Second, for each radius r_i we can randomly sample two disks a_i and b_i – if we repeat this enough times, one such pair is close to, in terms of rank, the leftmost and rightmost disks of radius r_i in the optimal solution. Last, as in the case of unit disks, we compute a $(1-\varepsilon)$-approximate clique in the resulting co-bipartite graph using the algorithm from Section 2.

We note here that our result for unit disk graphs is not subsumed by the one for t distinct radii since the latter gives a worse dependency on ε for $t = 1$.

1.1 Definitions and notation

Let $\mathcal{D}$ be a set of n disks in the plane, and let $G(\mathcal{D})$ be the *intersection graph* or *disk graph* of $\mathcal{D}$ where

$$G(\mathcal{D}) = (\mathcal{D}, \{\{D, D'\} \mid D, D' \in \mathcal{D}, D \neq D', \text{ and } D \cap D' \neq \emptyset\}).$$

A *unit disk graph* is a disk graph where all disks have unit radius, which we will assume to be 1. We will focus on unit disk graphs and on disk graphs where the disks have t different radii, for some given $t > 1$.

Let $D(c, r)$ be a disk with center c and radius r. If the radius is clear from the context, we may simply write $D(c)$. We refer to the intersection of two disks $D(c, r) \cap D(c', r')$ as their *lens*. The line through c and c' (oriented towards c') splits this lens into two *half-lenses*; a region left of this line and a region right of this line. See Figure 1.

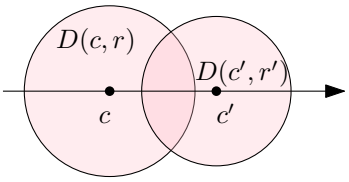

Figure 1 The lens $D(c, r) \cap D(c', r')$ of two intersecting disks $D(c, r), D(c', r')$.

Let a and b be two points in the plane. The Euclidean distance between a and b is denoted by $\text{dist}(a, b)$. The *vertical slab* of a and b is the region between the vertical lines through a and b, that is, the set of all points whose x-coordinates lie between the x-coordinates of a and b.

A graph is *co-bipartite* if its complement is bipartite. Let X and Y be sets of disks such that $G(X)$ and $G(Y)$ are both cliques. As observed in [15], $G(X \cup Y)$ is co-bipartite. Let A be a set. We denote its cardinality by $|A|$. Let $X^* \subseteq \mathcal{D}$ be a maximum-size clique in $G(\mathcal{D})$,

i.e., a subset of disks that all pairwise intersect. A clique $X \subseteq \mathcal{D}$ of size at least $(1-\varepsilon)|X^*|$ is a $(1-\varepsilon)$-approximate (maximum) clique.

2 Approximating the Maximum Clique in a co-Bipartite Disk Graph

In this section, we give a near-linear-time algorithm for computing a $(1-\varepsilon)$-approximate clique in a co-bipartite disk graph $G := G(X \cup Y)$. In such a graph, every $x, x' \in X$ and every $y, y' \in Y$ are adjacent, while $x \in X$ and $y \in Y$ are adjacent if and only if the distance between the corresponding centres is at most the sum of the corresponding radii. We begin with the straightforward transformation: a clique in G corresponds to an independent set in the complement of $\overline{G}$, which is bipartite. Thus, it is enough to compute a $(1-\varepsilon)$-approximate independent set in a bipartite graph $\overline{G}$. In this graph, we have an edge (x, y), for $x \in X$ and $y \in Y$, if and only if $\text{dist}(x, y) > r(x) + r(y)$, where $r(\cdot)$ denotes the radius and $\text{dist}(\cdot, \cdot)$ is the distance between the centres.

We first focus on computing the *cardinality* of a $(1-\varepsilon)$-approximate independent set in $\overline{G}$. Recovering the corresponding independent set is a bit more involved and will be explained later. For any graph, the complement of an independent set is a vertex cover. Denoting by n the number of nodes in $\overline{G}$, this immediately gives us that the cardinality of the maximum independent set I^* is equal to n minus the cardinality of the minimum vertex cover C^*. We first observe that a similar property holds for their approximations.

▶ **Proposition 1.** *Let c be an $(1+\varepsilon)$-approximation of $|C^*|$, that is, $|C^*| \le c \le (1+\varepsilon)|C^*|$. Then, $n - c$ is an $(1-\varepsilon)$-approximation of $|I^*|$, that is, $(1-\varepsilon)|I^*| \le n - c \le |I^*|$.*

Proof. Because $|C^*| = n - |I^*|$, we have $n - c \le |I^*|$. We observe that $|I^*| \ge n/2$, as one of the sides of $\overline{G}$ consists of at least $n/2$ nodes forming an independent set. Then, $n - c \ge n - (1+\varepsilon)|C^*| = n - (1+\varepsilon)(n - |I^*|) = |I^*| - \varepsilon \cdot n + \varepsilon|I^*| \ge (1-\varepsilon)|I^*|$. ◀

We now focus on obtaining a $(1+\varepsilon)$-approximation of $|C^*|$. For a bipartite graph, Kőnig's theorem states that the cardinality of the minimum vertex cover is equal to the cardinality of the maximum matching. Thus, denoting by M^* the maximum matching, it is enough to compute a $(1-\varepsilon/2)$-approximation of $|M^*|$, denoted m. Indeed, we can assume $\varepsilon \le 1$, and then $(1-\varepsilon/2)|M^*| \le m \le |M^*|$ implies $|M^*| \le m/(1-\varepsilon/2) \le |M^*|/(1-\varepsilon/2) \le (1+\varepsilon)|M^*|$, so $m/(1-\varepsilon/2)$ can be returned as a $(1+\varepsilon)$-approximation of $|C^*|$.

To design the algorithm, we need an efficient data structure for maintaining a dynamic set of points $S \subseteq \mathbb{R}^2$. Each $s \in S$ has its associated weight $w_s \in \mathbb{R}$, and we define a distance function $\delta : \mathbb{R}^2 \times S \to \mathbb{R}$ as $\delta(p, s) = w_s + \text{dist}(p, s)$. The query returns, for a given point $x \in \mathbb{R}^2$, the furthest point in S, that is, $\arg\max_{s \in S} \delta(x, s)$. This can be seen as a point location query in the furthest site additively weighted Voronoi diagram of S.

▶ **Lemma 2.** *We can maintain a dynamic set of of weighted points S of size n such that an insertion takes amortized expected time $O(\log^2 n)$, a deletion takes amortized expected time $O(\log^4 n)$, and so that a point location query in the furthest site additively weighted Voronoi diagram of S can be answered in worst-case deterministic time $O(\log^2 n)$.*

Proof. As explained by Agarwal, Efrat, and Sharir [2, Section 9.1], the problem reduces to maintaining the upper envelope of a set of n totally defined continuous bivariate functions of constant description complexity in three dimensions, under insertions, deletions, and vertical ray shooting queries. The discussion of Agarwal, Efrat, and Sharir [2, Section 9.1] also shows that the complexity of these upper envelopes (and thus of the corresponding farthest-site

Voronoi diagrams) is linear. Thus, the result follows from the data structure given in [27, Theorem 8.3] and its subsequent improvement [31]. ◀

With the above data structure in hand, we can describe the algorithm for computing a $(1-\varepsilon)$-approximate matching in $\overline{G}$. As explained above, returning n minus its cardinality gives us the cardinality of a $(1-\varepsilon)$-approximate clique in G.

▶ **Lemma 3.** *Let G be a co-bipartite disk graph on n vertices. A $(1-\epsilon)$-approximate matching of $\overline{G}$ can be computed in expected $O((n/\varepsilon)\log^4 n)$ time.*

Proof. Let the given co-bipartite graph be $G(X \cup Y)$. We will call X the *left side*, and Y the *right side*. We recall that the algorithm of Hopcroft and Karp [24] proceeds in phases while maintaining the current matching M (initially empty). In the beginning of each phase, we run a breadth-first search to partition the nodes of the graph into *layers*. We start the search from the unmatched nodes on the left side of the graph that form layer 0. On even layers of the search, we can traverse any unmatched edge. On odd layers of the search, we can only traverse a matched edge. The search terminates after the first layer containing an unmatched node on the right side of the graph (or when no further layers can be created, meaning that the current matching M is a maximum matching). This gives us the length k of a shortest augmenting path. Next, the algorithm finds a maximal set of node-disjoint augmenting paths of length k. This can be implemented by running a depth-first search from the unmatched nodes in the last layer. The depth-first search is only allowed to follow an edge leading to a yet unvisited node in the previous layer, and the paths in its tree must alternate between matched and unmatched edges, as in the breadth-first search.

The breadth-first search and the depth-first search run in time linear in the size of the graph, and the length of a shortest augmenting path increases by at least 2 in every phase. It is well-known that if the shortest augmenting path is of length at least $2/\varepsilon$ then the current matching M is a $(1-\varepsilon)$-approximate maximum matching. Thus, it is enough to execute $1/\varepsilon$ phases to obtain such a matching. However, the number of edges in our graph might be even quadratic in n, so to finish the proof we need to show how to implement a single phase in $O(n\log^4 n)$ expected time.

We first explain how to implement the breadth-first search in expected $O(n\log^4 n)$ time. To this end, we maintain an instance of the data structure from Lemma 2, initially containing the set S of all points corresponding to the nodes on the right side of the graph. We set $w_s = -r$ for a point $s \in S$ corresponding to a disk of radius r. Then, on every even layer, we iterate over the nodes there, and for each such node u corresponding to a disk $D(c,r)$, we need to iterate over the yet unvisited nodes v such that (u,v) is an edge. This is equivalent to obtaining (and removing) all points $s \in S$ such that $\delta(c,s) > r$, which in turn can be implemented by repeatedly retrieving $\arg\max_{s\in S}\delta(c,s)$, checking if $\delta(c,s) > r$, and if so, removing s from S and repeating. On every odd layer, we do not need the data structure, as each node has at most one incident matched edge. The overall number of operations on the data structure is $O(n)$, and the deletions dominate, so the total time for the breadth-first search is as claimed. For the depth-first search, we maintain a separate instance of the data structure from Lemma 2 for each even layer ℓ, initially containing the set S_ℓ of all points corresponding to its nodes. Then, if we are currently at node v on an odd layer ℓ, we need to iterate over the yet unvisited nodes u from layer $\ell-1$ such that uv is an edge. Let the disk corresponding to v be $D(c,r)$, Then, this is again equivalent to retrieving (and removing) all points $s \in S_{\ell-1}$ such that $\delta(c,s) > r$, and it can be implemented as above. The overall number of operations on all data structures is $O(n)$, making the total time as required. ◀

For the case of unit disk graphs, Lemma 3 can be improved. For this, we just need a data structure that plays the role of the one in Lemma 2 and that is optimized for unit disk graphs. The following lemma shows how to do this, adapting a method by Efrat, Itai, and Katz [17] to our setting:

▶ **Lemma 4.** *Let S be a set of n unit disks. Then, we can preprocess S in $O(n \log n)$ time into a data structure that supports the following operations:*

- *find(q): given $q \in \mathbb{R}^2$, find a disk in the data structure that does not contain q, or report that no such disk exists (i.e., q lies in the intersection of all the disks in the data structure); and*
- *delete(s): delete the unit disk s from the data structure.*

The time for a find-operation is $O(\log n)$, and the total time to delete all the disks from the data structure is $O(n \log n)$.

Proof sketch. Our data structure is very similar to the structure by Efrat, Itai, and Katz [17, Section 5.1], and we refer the reader to their paper for further details.

Let $\mathcal{G}$ be a regular grid with cells of edge length $1/2$. During the preprocessing phase, we locate for every disk center in S the cell of $\mathcal{G}$ that contains it. Let C be a nonemtpy grid cell, and let S_C be the unit disks whose centers lie in C. Since the diameter of C is less than 1, the disks in S_C have non-empty intersection (e.g., the center of C must lie in all the disks of S_C). We compute the intersection P_C of S_C. The intersection P_C is a convex polygon-like structure whose boundary consists of arcs from S_C, where every disk from S_C contributes at most one arc. The arcs appear along the upper and the lower boundary of P_C in reverse order of the x-coordinates of the corresponding disk centers. Thus, the total complexity of P_C is $O(|S_C|)$, and it can be computed in time $O(|S_C| \log |S_C|)$, using a divide-and-conquer strategy. As in the data structure by Efrat, Itai, and Katz, we compute for each P_C two binary trees that represent the structure of the upper and the lower boundary of P_C, respectively, as in the well-known data structure by Overmars and van Leeuwen [33]. In these trees, the leaves correspond to the disks from S_C, in reverse x-order of their centers. The inner nodes represent the upper (or the lower) boundary of the intersection of the disks in the corresponding subtree. The boundary of each inner node is obtained by intersecting the boundaries of the two children. These boundary curves intersect in a single point. With each inner node, we store the intsersection point of the boundary curves of its children, as well as the part of the boundary curve for this node that does not appear on the parent boundary . The binary trees can be computed in $O(n \log n)$ time, and the total time for the preprocessing phase is thus $O(n \log n)$ (see the paper by Efrat, Itai, and Katz for the details).

A find-query can be answered as follows: given $q \in \mathbb{R}^2$, we first check if there is a non-empty grid cell C that is far enough from q such that q must lie outside of P_C. If such a cell C exists, we report an arbitrary disk from P_C and are done (this takes $O(\log n)$ time, assuming that the nonempty grid cells are stored in a suitable data structure). Otherwise, there is a constant number of remaining candidate cells, and for each such candidate cell C, we determine whether q lies outside of P_C. This can be done using the tree structures for C to determine whether q lies above or below the upper and the lower boundary of P_C, by descending into the binary tree, using the intersection points of the child boundaries as a guidance. If q lies outside of P_C, we can use the result of the tree-searches to obtain a disk in S_C that does not contain q, as desired. Again, this takes $O(\log n)$ time.

To delete a a disk s from S, we find the cell C that contains the center of s, and we delete s from S_C. For this, we must update the tree structures that represent the intersection P_C. The details for the algorithm and its analysis are given by Efrat, Itai, and Katz. Briefly,

the idea is to find the parts of the boundary that are occluded by the deleted disk, and to promote these parts up in the tree. The main observation is that every piece of the boundary can rise only upwards into the tree, and that the tree has only $O(\log n)$ levels, resulting in a total deletion time of $O(n \log n)$. ◀

Using Lemma 4, we get the following improved version of Lemma 3 for unit disk graphs:

▶ **Lemma 5.** *Let G be a co-bipartite unit disk graph on n vertices. A $(1-\epsilon)$-approximate matching of $\overline{G}$ can be computed in expected $O((n/\varepsilon)\log n)$ time.*

Proof. The proof is identical to the proof of Lemma 3, but using the data structure from Lemma 4 instead of Lemma 2. ◀

Lemma 3 and Lemma 5 allow us to compute the cardinality of the sought $(1-\varepsilon)$-approximate clique. Computing the corresponding subset of nodes is a bit more involved.

▶ **Theorem 6.** *Let G be a co-bipartite disk graph with n vertices. We can compute a $(1-\varepsilon)$-approximate clique of G in expected $O((n/\varepsilon)\log^4 n)$ time. When G is a unit disk graph, such a clique can be computed in expected $O((n/\varepsilon)\log n)$ time.*

Proof. We provide the analysis for general disk graphs. The case of unit disk graphs is handled in the same way, using the faster data structure from Lemma 4.

Our strategy is to find a $(1+\varepsilon)$-approximate vertex cover in $\overline{G}$ in $O((n/\varepsilon)\log^4 n)$ expected time, and we return its complement. By Proposition 1, this constitutes a $(1-\varepsilon)$-approximate clique in G.

To obtain the approximate vertex-cover, we proceed as follows: recall that X and Y are the nodes on the left and on the right side of G. As in the proof of Lemma 3, we start with executing $\ell = 1/\varepsilon$ phases of the Hopcroft-Karp algorithm. This takes expected time $O((n/\varepsilon)\log^4 n)$, and it produces a $(1-\varepsilon)$-approximate maximum matching M. Next, we again run a breadth-first search to partition the nodes into layers. Now, however, we do not need to terminate the search after reaching the first layer containing an unmatched node of Y. The BFS gives us a partition of X and Y into subsets

$$X_0, Y_0, X_1, Y_1, X_2, Y_2, \ldots$$

corresponding to the consecutive layers, where $X_0, X_1, \cdots \subseteq X$, and $Y_0, Y_1, \cdots \subseteq Y$, as well as the remaining part $X' = X \setminus \bigcup_i X_i$ and $Y' = Y \setminus \bigcup_i Y_i$. Because we have run ℓ phases of the Hopcroft-Karp algorithm, the layers $Y_0, Y_1, \ldots, Y_{\ell-1}$ do not contain any unmatched nodes, and hence $\sum_i |Y_i| = |M|$. We choose $i \in \{0, 1, \ldots, \ell-1\}$ such that $|Y_i| \leq |M|/\ell$, and we define the following set K:

$$K := (X \setminus (X_0 \cup X_1 \cup \cdots \cup X_i)) \cup (Y_0 \cup Y_1 \cup \cdots \cup Y_i).$$

We will now establish that K is a vertex cover for G, with size $(1+\varepsilon)|M|$.

Consider an edge $e = xy$, where $x \in X$ and $y \in Y$. We need to show that $x \in K$ or $y \in K$. If $x \in K$, we are done. Thus, suppose that $x \notin K$, and we need to establish that $y \in K$. Since $x \notin K$, we have $x \in X_j$, for some $j \in \{0, 1, \ldots, i\}$. We consider two cases.

- **Case 1:** $e \in M$: Because $x \in X_j$ is matched, we must have $j \geq 1$ and we have reached x through an augmenting path ending with e. Thus, $y \in Y_{j-1} \subseteq K$.
- **Case 2:** $e \notin M$: We can extend an augmenting path ending at x with e to obtain an augmenting path ending at y. Thus, $y \in Y_0 \cup Y_1 \cup \cdots \cup Y_j \subseteq K$.

This shows that K is a vertex cover.

We now move to analyzing the size of K. First, we argue that every node in K is an endpoint of an edge in M. We prove this separately for the nodes in X and in Y.

- **Case 1:** $x \notin X_0 \cup X_1 \cup \cdots \cup X_i$**:** Such a node is matched, as it does not belong to X_0.
- **Case 2:** $y \in Y_0 \cup Y_1 \cup \cdots \cup Y_i$**:** None of the sets $Y_0, Y_1, \ldots, Y_{\ell-1}$ contain an unmatched node, so such a node is matched.

Next, we consider the situation when an edge $xy \in M$ has two endpoints in K. Then, $x \notin X_0 \cup X_1 \cup \cdots \cup X_i$ and $y \in Y_j$, for some $j \in \{0, 1, \ldots, i\}$. Now, it is possible to extend an augmenting path ending at y by the matched edge yx to obtain an augmenting path ending at x, so $x \in X_{j+1}$. Thus, $j = i$, and the considered matched edge connects $x \in X_{i+1}$ and $y \in Y_i$. By the choice of i, there are at most $|M|/\ell$ such edges. This allows us to bound the number of nodes in K by $|M| + |M|/\ell = (1 + \varepsilon)|M|$.

Together, this shows that K is an $(1 + \varepsilon)$-approximate vertex cover.

As in Lemma 3, running the breadth-first search takes $O(n \log^4 n)$ expected time. After that, constructing K can easily be done in additional linear time. ◀

3 Unit Disk Graphs

We describe an algorithm that computes an $(1 - \varepsilon)$-approximate clique in a unit disk graph with probability at least $1 - \delta$ in $O((n/\varepsilon^2) \log(n) \log(1/\delta))$ expected time. Let P be a set of n points in the plane, and let $\mathcal{D}(P) = \{D(p, 1) \mid p \in P\}$ be the set of unit disks whose centers are given by P. For convenience, we will sometimes identify disks with their centers. A set $X \subseteq P$ is said to form a clique if all the disks in $\mathcal{D}(X)$ intersect pairwise.

Let $\mathcal{G}$ be a regular grid where each grid cell has diameter 2, and let $\mathcal{C}$ be the collection of grid cells that contain at least one point from P. Let $C \in \mathcal{C}$ be such a nonempty cell. The *extension* of C, denoted by C^+, is the 5×5-block of grid cells whose central cell is C. Furthermore, let $P_C \subseteq P$ be the set of disk centers that lie in any cell of the extension C^+; for an illustration, see Figure 2. Each point of P_C is uniquely assigned to exactly one cell of C^+: if a point lies on the boundary shared by multiple cells of C^+, it is assigned arbitrarily to one of them. Then, a simple diameter argument shows that every clique in P must be contained within the extension of some nonempty cell.

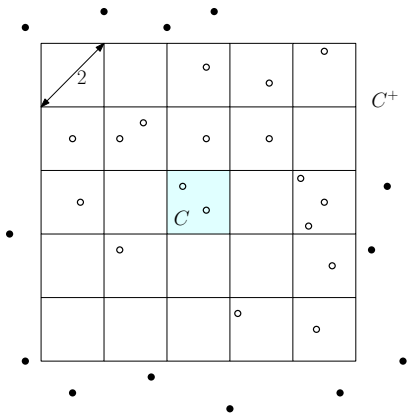

Figure 2 A cell C (the central cell) and its extension C^+, where each cell in C^+ has a diameter 2. All points (both solid and hollow) belong to P, while the hollow points also belong to P_C.

▷ Claim 7. Let X be a clique in P. Then, there exists a cell $C \in \mathcal{C}$ such that $X \subseteq P_C$.

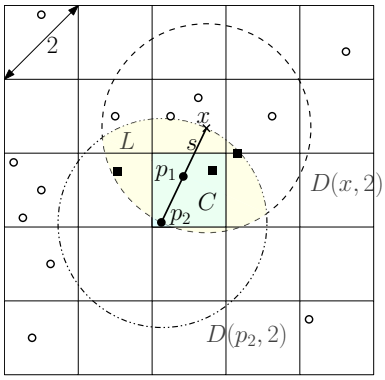

Figure 3 The cell C is represented by the light-blue colored cell with $p_1, p_2 \in C$. The thick black line denotes the line segment $s = \overline{p_2 x}$ of length 2, passing through p_1. The dotted disk represents $D(x, 2)$, while the dash-dotted disk represents $D(p_2, 2)$. The yellow-shaded region corresponds to the lens $L = D(x, 2) \cap D(p_2, 2)$, formed along the axis s.

Proof. Let $p \in X$, and let $C \in \mathcal{C}$ be the cell that contains p. Since every other disk center in X has distance at most 2 from p, it follows that X must lie in C^+. Hence, we have $X \subseteq P_C$, as claimed. ◀

Furthermore, a simple packing argument shows that every extension of a nonempty cell C contains a clique of size linear in $|P_C|$.

▷ Claim 8. Let X^* be a maximum clique in P_C. Then, we have $|X^*| \geq |P_C|/25$.

Proof. The extension C^+ consists of 25 grid cells with pairwise disjoint interiors, where each cell has diameter 2. Hence, for any such cell $C' \subset C^+$, the set $P_C \cap C'$ must form a clique, since all these disks are pairwise intersecting. Recall that if a disk center lies on the boundary of a cell, it is assigned arbitrarily to one of the adjacent cells so that each point in P_C is uniquely assigned to exactly one cell in C^+. Since there are 25 such cells, there exists at least one cell in C^+ containing at least $|P_C|/25$ disk centers. It follows that P_C contains a clique of size at least $|P_C|/25$, and therefore the maximum clique X^* is at least of size $|P_C|/25$. ◀

Now, let $C \in \mathcal{C}$. We select two disk centers $p_1, p_2 \in P_C$ independently and uniformly at random. Let s be the line segment of length 2 that starts at p_2 and is directed toward p_1, so that p_1 lies either on s or on its extension along the same direction. Let x be the other endpoint of s, and let $L = D(x, 2) \cap D(p_2, 2)$ be the lens with axis s; see Figure 3 for an illustration. We determine the set $P_L = L \cap P_C$ of disks centers that lie in L, and we find an $(1 - \varepsilon/2)$-approximate maximum clique X for P_L, using Theorem 6 from Section 2. The lens L is partitioned by segment s into two parts: the part to the left of s and the part to the right of s. Without loss of generality, the points on s belong to the left part. Any two points in the same part have a distance at most 2. Thus, all disks with centers in the same part form a clique. In other words, the intersection graph of unit disks centered at P_L is a co-bipartite graph. Consequently, as we show below, the clique X is an $(1 - \varepsilon)$-approximate clique for P_C with probability $\Omega(\varepsilon)$.

▶ **Lemma 9.** *Let X^* be a maximum clique in P_C. Then, with probability $\Omega(\varepsilon)$, we have $|X| \geq (1 - \varepsilon)|X^*|$.*

Proof. First, by Claim 8, we know that $|X^*| \geq |P_C|/25$. Thus, with probability at least $1/25$, we have that $p_1 \in X^*$. Now, we sort the centers in X^* in the increasing order of their

distances from p_1, and let π^* be the resulting ordering of X^*. Let X^+ be those centers from X^* that have rank at least $(1-\varepsilon/2)|X^*|$ in π^*, and let $X^- = X^* \setminus X^+$ (in other words, X^+ contains the $(\varepsilon/2)|X^*|$ points in X^* that are furthest away from p_1, and X^- contains the $(1-\varepsilon/2)|X^*|$ remaining points). Since

$$|X^+| = \frac{\varepsilon}{2}|X^*| \geq \frac{\varepsilon}{50}|P_C|,$$

it follows that with probability at least $\varepsilon/50$, the second sampled center p_2 lies in X^+.

Since p_1 and p_2 are sampled independently, we can conclude that with probability at least $\varepsilon/1250$, it holds that $p_1 \in X^*$ and that $p_2 \in X^+$. Henceforth, we assume that indeed this is the case. Since p_1 and p_2 both lie in X^*, the distance $\operatorname{dist}(p_1, p_2)$ between p_1 and p_2 is at most 2. From this, it follows that $D(p_1, \operatorname{dist}(p_1, p_2)) \subseteq D(x, 2)$ since by the choice of x, we have $\operatorname{dist}(x, p_1) + \operatorname{dist}(p_1, p_2) = 2$, which implies that for every point $y \in D(p_1, \operatorname{dist}(p_1, p_2))$, the triangle inequality gives

$$\operatorname{dist}(x, y) \leq \operatorname{dist}(x, p_1) + \operatorname{dist}(p_1, y) \leq \operatorname{dist}(x, p_1) + \operatorname{dist}(p_1, p_2) = 2,$$

as needed.

As $p_2 \in X^+$, and since all the points in X^+ are at least as far away from p_1 as all the points in X^-, it follows that $X^- \subseteq D(p_1, \operatorname{dist}(p_1, p_2)) \subseteq D(x, 2)$. Moreover, since $X^- \subseteq X^*$ and $p_2 \in X^*$, we also have $X^- \subseteq D(p_2, 2)$. This implies that X^- lies in the lens L defined by $D(x, 2)$ and $D(p_2, 2)$. In particular, P_L contains a clique of size $(1-\varepsilon/2)|X^*|$, and since X is a $(1-\varepsilon/2)$-approximate clique for P_L, we have

$$|X| \geq (1-\varepsilon/2)(1-\varepsilon/2)|X^*| = (1-\varepsilon+\varepsilon^2/4)|X^*| \geq (1-\varepsilon)|X^*|.$$

This means that X is an $(1-\varepsilon)$-approximate maximum clique for P_C, as claimed. ◀

We can now present the complete algorithm.

▶ **Theorem 10.** *Let P be a set of n points in the plane, let $\varepsilon > 0$ be a parameter. There is a randomized algorithm that runs in expected time $O((n/\varepsilon^2)\log n)$ and computes an $(1-\varepsilon)$-approximate clique for $\mathcal{D}(P)$ with probability at least $1/2$.*

Proof. Let $C \in \mathcal{C}$ be a nonempty grid cell. By Lemma 9, and using Theorem 6, one can compute, in $O((|P_C|/\varepsilon)\log n)$ expected time, a clique for P_C that, with probability $\Omega(\varepsilon)$, is an $(1-\varepsilon)$-approximate clique for P_C. By repeating this process $O(1/\varepsilon)$ times, and selecting the largest of the resulting cliques, one can compute in $O((|P_C|/\varepsilon^2)\log n)$ expected time, a clique that is an $(1-\varepsilon)$-approximate clique for P_C with probability at least $1/2$.

We do this for P_C of every nonempty cell $C \in \mathcal{C}$, and we return the largest of the resulting cliques. (Note that all non-empty cells can be easily found in $O(n \log n)$ time [36]). By Claim 7, there exists a cell $C \in \mathcal{C}$ such that P_C contains a max-clique of P, so with probability at least $1/2$, this gives an $(1-\varepsilon)$-approximate maximum clique for P. The total running time is proportional to

$$\sum_{C \in \mathcal{C}} \frac{|P_C|}{\varepsilon^2}\log n = \frac{\sum_{C \in \mathcal{C}}|P_C|}{\varepsilon^2}\log n = O((n/\varepsilon^2)\log n),$$

since every point from P appears in $O(1)$ sets P_C. The theorem follows. ◀

By a standard argument, for any value $\delta > 0$, we can raise the probability of success to $1-\delta$ by independently repeating the algorithm of Theorem 10 $O(\log(1/\delta))$ times and by taking the best solution. This increases the running time by an $O(\log(1/\delta))$ factor.

4 Disk graphs with different radii

We first review the algorithm by Keil and Mondal [28]. The main idea is simple and elegant, and it goes as follows. Suppose that there are t distinct radii $r_1, \ldots, r_t$, in increasing order. One can enumerate all radii that appear in a maximum clique and, for each radius r_i, enumerate all disk pairs to find the one with centers a_i, b_i that are, respectively, leftmost and rightmost (in the optimum) along the x-axis. Let X be the set of these $2t$ disks. For each pair of centers a_i, b_i, consider their vertical slab and find the set L_i of all radius-r_i disks that intersect all disks in X and have their centers within the slab and above the line segment a_ib_i. Similarly, find the set R_i of all radius-r_i disks that intersect all disks in X and have their centers in the slab and below the line segment a_ib_i. Take $L = L_1 \cup L_2 \cup \cdots \cup L_t$ and $R = R_1 \cup R_2 \cup \cdots \cup R_t$. A simple geometric argument shows that the graph induced by $L \cup R$ is co-bipartite, and thus, similar to the case of unit disks, one can find a maximum clique by finding a maximum matching. The algorithm runs in time $O(2^t n^{2t}(f(n) + n^2))$, where $f(n)$ is the time to compute a maximum matching in an n-vertex bipartite graph. The factor n^{2t} comes from the enumeration to find the leftmost and rightmost disks in the optimal solution for every radius r_i.

In this section, we give an efficient parameterized approximation scheme w.r.t. the number t of different radii and ε. There are two places where we can speed up the algorithm described above if approximate solutions are allowed. First, instead of finding the leftmost and rightmost disks for each radius by enumeration, we use random sampling to find two centers a_i, b_i that are close (in terms of rank) to the leftmost and rightmost disks in the optimal clique. Second, as in the case of unit disk graphs, we can replace finding a maximum clique by finding an approximate maximum clique using Theorem 6.

Let $\mathcal{D}_i$ be the set of disks of radius r_i and let D^* be a maximum clique, with $|\mathcal{D}^*| = k^*$. In the next subsection, we will assume that $\mathcal{D}^*$ contains at least εk disks from each $\mathcal{D}_i$, for some parameter $k \in \mathbb{N}$. Later, in Section 4.2, we will drop this assumption using brute-force.

4.1 A special case: Disks of every radius contribute to the maximum clique

In this section, we will compute an $(1-\varepsilon)$-approximate maximum clique that has at least one disk from each radius.

Our algorithm is simple: we (independently) pick a disk $D(o, r_1)$ uniformly at random $m_1 = \Theta(n/k\varepsilon)$ times and compute the subset of disks $\mathcal{D}'$ that intersect $D(o, r_1)$, where r_1 is the smallest radius. With each choice of $D(o, r_1)$ and the corresponding $\mathcal{D}'$, we repeat the following procedure $m_2 = \Theta((k^*/k\varepsilon^2)^{2t})$ times:

1. For each radius r_i: uniformly at random pick two disks $D(a_i, r_i)$ and $D(b_i, r_i)$ from $\mathcal{D}'_i = \mathcal{D}_i \cap \mathcal{D}'$. Let X denote the set of the selected disks for all r_i. Without loss of generality, assume that a_i is to the left of b_i. We interpret a_i as the leftmost disk of radius r_i in the clique, and b_i as the rightmost disk in the clique.
2. Compute the subset of disks $\mathcal{D}'' \subseteq \mathcal{D}'$ that intersect all disks in X (by explicitly checking if a disk $D(c, r) \in \mathcal{D}'$ intersects all disks in X).
3. Form two sets of disks L and R. For each vertical slab with a_i, b_i on the boundary, find all the radius-r_i disks $L_i \subseteq \mathcal{D}''$ with center within the slab and above the line segment a_ib_i. Similarly, find all the radius-r_i disks $R_i \subseteq \mathcal{D}''$ with centers in the slab and below the line segment a_ib_i. Take $L = L_1 \cup L_2 \cup \cdots \cup L_t$ and $R = R_1 \cup R_2 \cup \cdots \cup R_t$.

4. Use the algorithm from Theorem 6 to compute an $(1-\varepsilon/2)$-approximate clique in $G(L \cup R)$. Note that this is possible, as $G(L \cup R)$ is co-bipartite, see Lemma 12 below.

We finally return the largest clique among all the repeated $m_1 m_2$ runs.

For the analysis, we start with the following observation.

▶ **Lemma 11.** *For a disk $D(o, r_1)$, let $\mathcal{D}'$ be the set of disks that intersect $D(o, r_1)$, Then, $|\mathcal{D}'| \leq 6k^*$.*

Proof. Consider the six cones of angle $\pi/3$ that partition $\mathbb{R}^2$ and have their apex at o. Any two disks $D(c_i, r_i)$ and $D(c_j, r_j)$ with the centers in the same cone and such that both intersect $D(o, r_1)$ must also pairwise intersect. To see that, we have $\text{dist}(o, c_i) \leq r_1 + r_i$ and $\text{dist}(o, c_j) \leq r_1 + r_j$. We also have that $\angle c_i o c_j \leq \pi/3$. Thus, $\text{dist}(c_i, c_j) \leq \max\{\text{dist}(o, c_i), \text{dist}(o, c_j)\} \leq r_i + r_j$, since $r_1 \leq r_i, r_j$. Hence, the disks from $\mathcal{D}'$ that are in a single cone form a clique. There must be a cone containing at least $|\mathcal{D}'|/6$ centers from $\mathcal{D}'$, and, consequently, we have $k^* \geq |\mathcal{D}'|/6$. ◀

The probability that by random sampling one selects a specific disk $D(o, r_1)$ in $\mathcal{D}^*$ is at least $\varepsilon k/n$. Since we repeat the algorithm $m_1 = \Theta(n/k\varepsilon)$ times, with a constant probability $D(o, r_1) \in \mathcal{D}^*$. In the following, we assume that $D(o, r_1)$ indeed appears in $\mathcal{D}^*$, and then use a similar argument as in the unit disk case to argue that from each radius r_i, we discard at most an $\varepsilon/2$-fraction of the disks from $\mathcal{D}^* \cap \mathcal{D}_i$ in our clique.

Let $c_1, \ldots, c_h$ be the centers of the disks in $\mathcal{D}^* \cap D_i$, ordered from left to right, and note that by our assumption on $\mathcal{D}^*$, we have $h \geq \varepsilon k$. By Lemma 11, the probability that $a_i \in \{c_1, \ldots, c_{(\varepsilon/4)h}\}$ is

$$\frac{(\varepsilon/4)h}{|D_i'|} \geq \frac{(\varepsilon/4)h}{|D'|} \geq \frac{\varepsilon h}{24k^*} \geq \frac{\varepsilon^2 k}{24k^*}.$$

Similarly, the probability that b_i is among the rightmost $(\varepsilon/4)h$ centers of $\mathcal{D}^* \cap \mathcal{D}_i$ is also at least $\varepsilon^2 k/24k^*$.

The following technical lemma, stated for our setting, asserts that we include at least $h - 2(\varepsilon/4)h = (1 - \varepsilon/2)h$ disks from $\mathcal{D}^* \cap \mathcal{D}_i$ in $L \cup R$.

▶ **Lemma 12.** *Assume that $D(o, r_1)$ and $D(a_i, r_i)$, $D(b_i, r_i)$, for all i are from $\mathcal{D}^*$. The disks in L form a clique. Similarly, the disks in R form a clique. Further, for every i, all disks in $\mathcal{D}^*$ of radius r_i with centers within the vertical slab within a_i and b_i are included in $L \cup R$.*

Proof. The proof that L and R form a clique respectively follows directly from [28, Lemma 3.1]. Next, all disks in D^* intersect $D(o, r_1)$, and $D(a_i, r_i), D(b_i, r_i)$, for all i. Thus the disks of radius r_i of D^* with the conditions specified in the lemma meet the requirements for L_i or R_i and thus are included in $L \cup R$. ◀

Hence, with probability at lest $(\varepsilon^2 k/24k^*)^2$ we include at least a $(1 - \varepsilon/2)$ fraction of the disks of $\mathcal{D}^* \cap D_i$ in $L \cup R$. The probability that we include $(1 - \varepsilon/2)$ fraction of all disks (e.g. over all t radii) is thus at least $(\varepsilon^2 k/24k^*)^{2t}$.

We now compute a $(1 - \varepsilon/2)$-approximate clique in $G(L \cup R)$, so the size is at least $(1 - \varepsilon/2)(1 - \varepsilon/2)k^* \geq (1 - \varepsilon)k^*$.

Repeating this $m_2 = \Theta((k^*/k\varepsilon^2)^{2t})$ times, the algorithm succeeds with a constant probability. If $k = \Theta(k^*)$, then $m_2 = O(1/\varepsilon^{4t})$.

Now we can conclude with the full algorithm and its runtime.

▶ **Lemma 13.** *With a constant success probability, we can compute a $(1-\varepsilon)$-approximate maximum clique of n disks of t distinct radii, in which every radius appears at least εk times, in expected time*

$$O\left(n\log n + \frac{n}{k\varepsilon}(t\log n + k^*) + \frac{nk^*}{k\varepsilon}\left(\frac{k^*}{k\varepsilon^2}\right)^{2t}\left(t + \frac{\log^4 k^*}{\varepsilon}\right)\right)$$

where k^* is the size of the maximum clique.

Proof. For each disk $D(o, r_1)$, finding the disks that intersect it can be done easily in $O(n)$ time. But if we repeat this step $m_1 = \Theta(n/k\varepsilon)$ times, this will be too costly. Instead, we pre-process the disks into a data structure and answer a 2D circular range query, which returns the points within a query disk. In particular, for each radius r_i we process the centers of the disks in $\mathcal{D}_i$ into a size $O(|\mathcal{D}_i|)$ data structure using the construction in [1] such that one can report all centers of the disks in $\mathcal{D}_i$ within distance $r_1 + r_i$ from o in $O(\log n + \ell_i)$ time, where ℓ_i is the size of the output. The construction time for the data structure for $\mathcal{D}_i$ is $O(|\mathcal{D}_i|\log|\mathcal{D}_i|)$. Thus total construction time for all t different radii is $O(n\log n)$. We answer tm_1 queries and the total query time is $O(tm_1\log n + m_1k^*)$.

Note that Steps 1–4 of our algorithm operate on $\mathcal{D}'$ for each choice of $D(o, r_1)$ with $|\mathcal{D}'| = O(k^*)$ disks. Thus, the running time for Steps 1–4 is $O(2tk^* + f_{\varepsilon/2}(k^*))$, where $f_\varepsilon(n)$ is the time for computing a $(1-\varepsilon)$-approximate clique in an n-by-n co-bipartite disk graph. Applying Theorem 6, we have $f_\varepsilon(n) = O((n/\varepsilon)\log^4 n)$ expected time, so we have one iteration of Steps 1–4 to be $O(tk^* + (k^*/\varepsilon)\log^4 k^*)$. Summing all the computation costs, we have,

$$O\left(n\log n + \frac{n}{k\varepsilon}(t\log n + k^*) + \frac{nk^*}{k\varepsilon}\left(\frac{k^*}{k\varepsilon^2}\right)^{2t}\left(t + \frac{\log^4 k^*}{\varepsilon}\right)\right). \qquad \blacktriangleleft$$

4.2 The complete algorithm

▶ **Theorem 14.** *Let $\mathcal{D}$ be a set of n disks, let t be the number of distinct radii in $\mathcal{D}$, and let $\varepsilon \in (0,1)$ be a parameter. With a constant success probability, we can compute a $(1-\varepsilon)$-approximate maximum clique in $G(\mathcal{D})$ in expected time $\tilde{O}\left((\frac{t}{\varepsilon})^{O(t)}n\right)$.*

Proof. First, we compute a 1/5-approximate maximum clique $\hat{D}$. This can be achieved in $O(n\log n)$ time as follows. It is well-known that every set of pairwise intersecting disks can be stabbed by four points [16, 34]. This implies that there exists a point that is covered by at least 1/4 of the disks in a maximum clique[5]. Thus, a point of maximum depth is covered by at least so many disks as well. Computing a point of maximum depth in an arrangement of disks is 3SUM-hard [4]; however, a point of $(1-\varepsilon)$-approximate maximum depth can be computed, with high probability, in $O((1/\varepsilon)^2 n\log n)$ expected time [4]. We therefore run this algorithm for $\varepsilon = 1/2$ and get the approximate clique of the desired size. Thus, we take $k = |\hat{D}|$, where $k^*/5 \le k \le k^*$.

Consider the optimal solution D^* with k^* disks and assume that all disks of radius r_i are removed from D^* if there are at most $\varepsilon k^*/(2t)$ of them in it. This leaves a clique $\bar{D} \subseteq D^*$ with $(1-\varepsilon/2)k^* \le |\bar{D}| \le k^*$. Note that $\bar{D}$ has at least $\varepsilon k^*/2t \ge \varepsilon k/2t$ of disks in radius r_i, if r_i appears at all. We now aim to find a $1-\varepsilon/5$ approximation to $\bar{D}$, which is a $1-\varepsilon$

[5] Carmi, Katz and Morin [10] showed that given a set of pairwise intersecting disks one can compute a set of four stabbing points in linear time but it's not obvious how to find a point as above in the same time.

approximation to D^*. Of course, we do not know in advance what those radii are that appear less than $\varepsilon k^*/(2t)$ in D^*, which is some (fixed but unknown) optimal solution, and therefore we will brute-force through all subsets of radii, at the cost of an extra factor of 2^t in the running time.

Let $\mathcal{R}$ denote the ordered set of distinct radii appearing in $\mathcal{D}$. For each subset R of radii, we run the algorithm from Lemma 13 with parameters $\varepsilon/2$ and k/t. We output the best solution. The correctness and the approximation factor follow immediately from Lemma 13 and the fact that we enumerate all radius subsets, one of which matches the set of radii that appear more than $\varepsilon k^*/(2t)$ times in the optimal solution D^*. For that particular run, Lemma 13 guarantees that we find a $1-\varepsilon$ approximation to D^* with a constant probability.

The running time comes from repeating Lemma 13 with parameters $\varepsilon/2$ and k/t, for 2^t times. Notice that the construction of the circular range query data structure is only done once at the initialization step. But all other steps will be repeated 2^t times. Further, $k^*/5 \leq k \leq k^*$. The total running time is obtained from applying Lemma 13.

$$O\left(n\log n + \frac{2^t t^2 n\log n}{k^*\varepsilon} + \frac{t2^t n}{\varepsilon} + \frac{2^{7t}t^{2t+1}n\log^4 k^*}{\varepsilon^{4t+2}} + \frac{nt2^{4t+1}}{\varepsilon^{4t+1}}\right) = \tilde{O}\left(\left(\frac{t}{\varepsilon}\right)^{O(t)} n\right).$$

◀

As in the case of unit disks, we can increase the probability of success to any desired value by independently repeating the algorithm sufficiently many times.

Acknowledgement

The work in this paper was initiated at the Lorentz Center workshop on Fine-Grained & Parameterized Computational Geometry, held in February 2025.

References

1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2009)*, pages 180–186, 2009. doi:10.1137/1.9781611973068.21.

2 Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 2000. doi:10.1137/S0097539795295936.

3 Christoph Ambühl and Uli Wagner. The clique problem in intersection graphs of ellipses and triangles. *Theory Comput. Syst.*, 38(3):279–292, 2005. doi:10.1007/S00224-005-1141-6.

4 Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008. doi:10.1137/060669474.

5 Marthe Bonamy, Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Panos Giannopoulos, Eun Jung Kim, Paweł Rzążewski, Florian Sikora, and Stéphan Thomassé. EPTAS and subexponential algorithm for maximum clique on disk and unit ball graphs. *J. ACM*, 68(2):1–38, 2021. doi:10.1145/3433160.

6 Édouard Bonnet, Nicolas Grelier, and Tillmann Miltzow. Maximum clique in disk-like intersection graphs. In *Proceedings of the 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *LIPIcs*, pages 17:1–17:18, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.17.

7 Heinz Breu. *Algorithmic Aspects of Constrained Unit Disk Graphs*. PhD thesis, Department of Computer Science, University of British Columbia, 1996. doi:10.14288/1.0051600.

8 Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry*, 9(1):3–24, 1998. Special Issue on Geometric Representations of Graphs. doi:10.1016/S0925-7721(97)00014-X.

9 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discret. Comput. Geom.*, 50(3):771–783, 2013. doi:10.1007/s00454-013-9538-5.

10 Paz Carmi, Matthew J Katz, and Pat Morin. Stabbing pairwise intersecting disks by four points. *Discret. Comput. Geom.*, 70:1751–1784, 2023. doi:10.1007/s00454-023-00567-0.

11 Timothy M Chan. Personal communication with the authors of [28], 2023.

12 Timothy M Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discret. Comput. Geom.*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

13 Li Chen, Rasmus Kyng, Yang Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *J. ACM*, 72(3):19:1–19:103, 2025. doi:10.1145/3728631.

14 Julia Chuzhoy and Sanjeev Khanna. Maximum bipartite matching in $n^{2+o(1)}$ time via a combinatorial algorithm. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, (STOC 2024)*, pages 83–94, 2024. doi:10.1145/3618260.3649725.

15 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discret. Math.*, 86(1):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

16 Ludwig W. Danzer. Zur Lösung des Gallaischen Problems über Kreisscheiben in der Euklidischen Ebene. *Stud. Sci. Math. Hungar.*, 21(1-2):111–134, 1986.

17 Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001. doi:10.1007/S00453-001-0016-8.

18 David Eppstein. Graph-theoretic solutions to computational geometry problems. In *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science, (WG 2009)*, volume 5911 of *LNCS*, pages 1–16, 2009. doi:10.1007/978-3-642-11409-0_1.

19 Jared Espenant, J. Mark Keil, and Debajyoti Mondal. Finding a maximum clique in a disk graph. In *Proceedings of the 39th International Symposium on Computational Geometry, (SoCG 2023)*, volume 258 of *LIPIcs*, pages 30:1–30:17, 2023. doi:10.4230/LIPICS.SOCG.2023.30.

20 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, (FOCS 1991)*, pages 2–12, 1991. doi:10.1109/SFCS.1991.185341.

21 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and Algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/A13060146.

22 Aleksei V. Fishkin. Disk graphs: A short survey. In *Proceedings of the 1st International Workshop on Approximation and Online Algorithms, (WAOA 2003)*, volume 2909 of *LNCS*, pages 260–264, 2003. doi:10.1007/978-3-540-24592-6_23.

23 Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier. In *Proceedings of the 18th Annual European Symposium on Algorithms, (ESA 2010), Part I*, volume 6346 of *LNCS*, pages 243–254, 2010. doi:10.1007/978-3-642-15775-2_21.

24 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973. URL: https://doi.org/10.1137/0202019.

25 Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. A unified approach to approximation schemes for NP- and PSPACE-hard problems for geometric graphs. In *Proceedings of the 2nd Annual European Symposium on Algorithms, (ESA 1994)*, volume 855 of *LNCS*, pages 424–435, 1994. doi:10.1007/BFB0049428.

26 Donggu Kang and James Payor. Flow rounding. *arXiv*, 2015. URL: `http://arxiv.org/abs/1507.08139`.

27 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discret. Comput. Geom.*, 64(3):838–904, 2020.

28 J. Mark Keil and Debajyoti Mondal. The maximum clique problem in a disk graph made easy. In *Proceedings of the 41st International Symposium on Computational Geometry, (SoCG 2025)*, volume 332 of *LIPIcs*, pages 63:1–63:16, 2025. `doi:10.4230/LIPICS.SOCG.2025.63`.

29 J. Mark Keil, Debajyoti Mondal, Ehsan Moradi, and Yakov Nekrich. Finding a maximum clique in a grounded 1-bend string graph. *J. Graph Algorithms Appl.*, 26(1):553–575, 2022. `doi:10.7155/JGAA.00608`.

30 Paul Koebe. Kontaktprobleme der konformen abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Kl.*, 88:141–164, 1936.

31 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM J. Comput.*, 51(3):723–765, 2022. `doi:10.1137/20M1388371`.

32 Tim Nieberg, Johann L. Hurink, and Walter Kern. Approximation schemes for wireless networks. *ACM Trans. Algorithms*, 4(4):49:1–49:17, 2008. `doi:10.1145/1383369.1383380`.

33 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981. `doi:10.1016/0022-0000(81)90012-X`.

34 Lajos Stachó. A solution of Gallai's problem on pinning down circles. *Mat. Lapok*, 32(1-3):19–47, 1981–84.

35 Anastasiia Tkachenko and Haitao Wang. Computing maximum cliques in unit disk graphs. In *Proceedings of the 37th Canadian Conference on Computational Geometry, (CCCG 2025)*, pages 283–291, 2025.

36 Haitao Wang. Unit-disk range searching and applications. *J. Comput. Geom.*, 14(1):343–394, 2023. `doi:10.20382/JOCG.V14I1A13`.