Approximating Densest Subgraph in Geometric Intersection Graphs

Sariel Har-Peled
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Saladi Rahul
Indian Institute of Science (IISc), Bangalore, India

Abstract

For an undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, with n vertices and m edges, the *densest subgraph* problem, is to compute a subset $S \subseteq \mathsf{V}$ which maximizes the ratio $|\mathsf{E}_S|/|S|$, where $\mathsf{E}_S \subseteq \mathsf{E}$ is the set of all edges of G with endpoints in S. The densest subgraph problem is a well studied problem in computer science. Existing exact and approximation algorithms for computing the densest subgraph require $\Omega(m)$ time. We present near-linear time (in n) approximation algorithms for the densest subgraph problem on *implicit* geometric intersection graphs, where the vertices are explicitly given but not the edges. As a concrete example, we consider n disks in the plane with arbitrary radii and present two different approximation algorithms.

As a by-product, we show a reduction from (shallow) range-reporting to approximate counting/sampling which seems to be new and is useful for other problems such as independent query sampling.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric intersection graphs, Densest subgraph, Range searching, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.43

Related Version *Full Version*: https://www.arxiv.org/abs/2405.18337

Funding Research of Saladi Rahul is funded by Walmart Center for Tech Excellence.

1 Introduction

Given an undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, the ***density*** of a set $S \subseteq V(\mathsf{G})$ is $|\mathsf{E}_S|/|S|$, where each edge in $\mathsf{E}_S \subseteq \mathsf{E}(\mathsf{G})$ has both its vertices in S. In the *densest subgraph problem*, the goal is to find a subset of $V(\mathsf{G})$ with the maximum density. Computing the densest subgraph is a primitive operation in large-scale graph processing, and has found applications in mining closely-knit communities [12], link-spam detection [18], and reachability and distance queries [14]. See [7] for a detailed discussion on the applications of densest subgraph, and [22] for a survey on the recent developments on densest subgraph.

Exact algorithms for densest subgraph. Unlike the (related) problem of computing the largest clique (which is NP-Hard), the densest subgraph can be computed (exactly) in polynomial time. Goldberg [19] show how to reduce the problem to $O(\log n)$ instances of s–t min-cut problem. Gallo et al. [17] improved the running time slightly by using parametric max flow computation. Charikar [10] presented an LP based solution to solve the problem, for which Khuller and Saha [21] gave a simpler rounding scheme.

Approximation algorithms for densest subgraph. The exact algorithms described above require solving either an LP or an s–t min-cut instance, both of which are relatively expensive to compute. To obtain a faster algorithm, Charikar [10] analyzed a 2-approximation algorithm


which repeatedly removes the vertex with the smallest degree and calculates the density of the remaining graph. This algorithm runs in linear time. After that, Bahmani et al. [6] used the primal-dual framework to give a $(1+\varepsilon)$-approximation algorithm which runs in $O\big((m/\varepsilon^2)\log n\big)$ time. The problem was also studied in the streaming model the focus is on approximation using bounded space. McGregor et al. [24] and Esfandiari et al. [16] presented a $(1+\varepsilon)$-approximation streaming algorithm using roughly $\widetilde{O}(|\mathsf{V}|)$ space. There has been recent interest in designing *dynamic* algorithms for approximate densest subgraph, where an edge gets inserted or deleted in each time step [8, 26, 15].

Geometric intersection graphs. The geometric intersection graph of a set $\mathcal{D}$ of n objects is a graph $G=(V,E)$, where each object in $\mathcal{D}$ corresponds to a unique vertex in V, and an edge exists between two vertices if and only if the corresponding objects intersect. Further, in an *implicit geometric intersection graph*, the input is only the set of objects $\mathcal{D}$ and *not* the edge set E, whose size could potentially be quadratic in terms of $|\mathsf{V}|$.

Unlike general graphs, the geometric intersection graphs typically have more structure, and as such computing faster approximation algorithms [4, 11], or obtaining better approximation algorithms on implicit geometric intersection graphs has been an active field of research.

One problem closely related to the densest subgraph problem is that of finding the maximum clique. Unlike the densest subgraph problem, the maximum clique problem is NP-hard for various geometric intersection graphs as well (for e.g., segment intersection graphs [9]). However, for the case of unit-disk graphs, an elegant polynomial time solution by Clark et al. [13] is known.

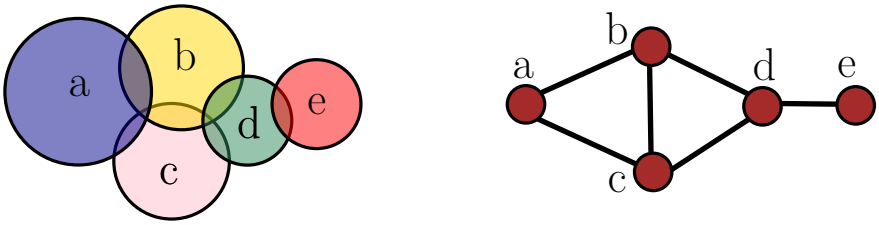

Figure 1.1 Five disks and their corresponding graph. The densest subgraph is $\{a,b,c,d\}$ with density $5/4$.

Motivation. Densest subgraph computation on geometric intersection graphs can help detect regions which have strong cellular network coverage, or have many polluting factories. The region covered (resp., polluted) by cellular towers (resp., or factories) can be represented by disks of varying radii.

1.1 Our results

Here, we study the densest subgraph problem on (implicit) geometric intersection graphs, and present near-linear time (in terms of $|\mathsf{V}|$) approximation algorithms.

From reporting to approximate counting/sampling. We show a reduction from (shallow) range-reporting to approximate counting/sampling: given a set of objects $\mathcal{D}$, and a reporting data-structure for $\mathcal{D}$, we show a reduction to building a data-structure, such that given a query object q, it returns approximately-at-uniform an object from $\mathsf{q}\sqcap\mathcal{D}=\{x\in\mathcal{D}\mid x\cap\mathsf{q}\neq\emptyset\}$, and also returns an $(1\pm\varepsilon)$-approximation for the size of this set. Previous work on closely

Approximation	Running time	Ref
$2+\varepsilon$	$O\left(\frac{n \log n}{\varepsilon^4}\right)$	Theorem 19
$1+\varepsilon$	$O\left(\frac{n \log^2 n}{\varepsilon^2}\left(\frac{1}{\varepsilon^2}+\log\log n\right)\right)$	Theorem 29

Figure 1.2 Our results.

related problem includes the work by Afshani and Chan [1] (that uses shallow counting queries), the work by Afshani et al. [2], and the work by Afshani and Philips [3]. For our application, this data-structure enables us to sample $(1 \pm \varepsilon)$-uniformly a disk from the set of disks intersecting a given disk. See Section 3 and Theorem 12 for details.

The reduction seems to be new, and should be useful for other problems. For example, in databases, motivated by summarizing large query output size, and to incorporate fairness and diversity in the output results, *independent query sampling (IQS)* has received quite a bit of attention. In IQS the goal is to report a uniform sample of the query output (i.e., of $\mathsf{q} \sqcap \mathcal{D}$). Crucially, the query output should be *independent* of all the previous queries posed (for example, if the same query is posed again, then the sample reported should be independent of the previous sample). See the survey by Tao [27] on IQS (and the references within). Our reduction addresses two of the future directions proposed in the survey: (a) designing a *generic* IQS data structure which works for a broad class of geometric queries, and (b) *approximate IQS* where the probability of reporting an object is almost-uniform, which is usually good enough in practice.

The application. For the sake of concreteness, we consider the case of n disks (with arbitrary radii) lying in the plane and present two different approximation algorithms. See Figure 1.1. However, our algorithms would work for other shapes with minor modifications.

A $(2+\varepsilon)$-approximation. Our first approximation algorithm uses the greedy strategy of removing disks of low-degree from the intersection graph. By batching the queries, and using the above data-structure, we get a $(2+\varepsilon)$-approximation for the densest subset of disks in time $O_\varepsilon(n \log n)$, where O_ε hides constants polynomial in $1/\varepsilon$. Getting this running time requires some additional ideas such as establishing densest subgraph's connection with deepest point in the arrangement of disks (defined later). The running time is optimal in terms of n in the comparison-based model, see Remark 20.

A $(1+\varepsilon)$-approximation. A more promising approach is to randomly sample edges from the intersection graph, and then apply known approximation algorithms. This requires some additional work since unlike previous work, we can only sample approximately in uniform. See Section 5 for details. The running time of the new algorithm is $O_\varepsilon(n \log^2 n \log\log n)$ which is slower than the first $(2+\varepsilon)$-approximation algorithm. The results are summarized in Figure 1.2.

2 Preliminaries

2.1 Definitions

In the following, $\mathcal{D}$ denotes a (given) set of n objects (i.e., disks). For $S \subseteq \mathcal{D}$ and $u \in \mathcal{D}$, we use the shorthands $S + u = S \cup \{u\}$ and $S - u = S \setminus \{u\}$. For $\varepsilon \in (0,1)$ and a real number $\alpha > 0$, let $(1 \pm \varepsilon)\alpha$ denote the interval $\big((1-\varepsilon)\alpha, (1+\varepsilon)\alpha\big)$. Throughout, a statement holds ***with high probability***, if it holds with probability at least $1 - n^{-c}$, where c is a sufficiently large constant.

▶ **Observation 1.**

(I) *For any ε, we have $\frac{1}{1+\varepsilon} \geq 1-\varepsilon$.*

(II) *For $\varepsilon \in (0,1/2)$, we have $\frac{1}{1\pm\varepsilon} = \big(1/(1+\varepsilon), 1/(1-\varepsilon)\big) \subseteq 1 \pm 2\varepsilon$ since $1-\varepsilon \leq \frac{1}{1+\varepsilon} \leq \frac{1}{1-\varepsilon} \leq 1+2\varepsilon$.*

(III) *For $\varepsilon \in (0,1/3)$, we have $(1\pm\varepsilon)^2 \subseteq (1\pm 3\varepsilon)$.*

(IV) *For $\varepsilon \in (0,1)$ and constants c, c_1 and c_2, such that $c \geq c_1 c_2$, we have $(1 \pm c_1\varepsilon/c)(1 \pm c_2\varepsilon/c) \subseteq 1 \pm \frac{c_1+c_2+1}{c}\varepsilon$*[1].

▶ **Definition 2.** *Given a set of objects $\mathcal{D}$ (say in $\mathbb{R}^d$), their* **intersection graph** *has an edge between two objects if and only if they intersect. Formally,*

$$\mathsf{G}_{\cap\mathcal{D}} = (\mathcal{D}, \{uv \mid u, v \in \mathcal{D}, \text{ and } u \cap v \neq \emptyset\}).$$

▶ **Definition 3.** *For a graph G, and a subset $S \subseteq \mathsf{V}(\mathsf{G})$, let*

$$\mathsf{E}_S = \mathsf{E}_S(\mathsf{G}) = \{uv \in \mathsf{E}(\mathsf{G}) \mid u, v \in S\}.$$

The **induced subgraph** *of G over S is $\mathsf{G}_S = (S, \mathsf{E}_S)$, and let $\mathsf{m}(S) = |\mathsf{E}_S|$ denote the number of edges in this subgraph.*

▶ **Definition 4.** *For a set $S \subseteq \mathsf{V}(\mathsf{G})$, its* **density** *in G is $\nabla(S) = \nabla_{\mathsf{G}}(S) = \mathsf{m}(\mathsf{G}_S)/|S|$, where $\mathsf{m}(\mathsf{G}_S)$ is the number of edges in G_S. Similarly, for a set of objects $\mathcal{D}$, and a subset $S \subseteq \mathcal{D}$, the* **density** *of S is $\nabla(S) = \mathsf{m}(\mathsf{G}_{\cap S})/|S|$.*

▶ **Definition 5.** *For a graph G, its* **max density** *is the quantity $\mathrm{d}(\mathsf{G}) = \max_{S\subseteq\mathsf{V}(\mathsf{G})} \nabla(S)$, and analogously, for a set of objects $\mathcal{D}$, its* **max density** *is $\mathrm{d}(\mathcal{D}) = \max_{S\subseteq\mathcal{D}} \nabla(S)$,*

The problem at hand is to compute (or approximate) the maximum density of a set of objects $\mathcal{D}$. If a subset S realizes this quantity, then it is the ***densest subset*** of $\mathcal{D}$ (i.e., $(\mathsf{G}_{\cap\mathcal{D}})_S$ is the densest subgraph of $\mathsf{G}_{\cap\mathcal{D}}$). One can make the densest subset unique, if there are several candidates, by asking for the lexicographic minimal set realizing the maximum density. For simplicity of exposition we threat the densest subset as being unique.

▶ **Lemma 6.** *Let $\mathcal{O} \subseteq \mathcal{D}$ be the densest subset, and let $\nabla = \nabla(\mathcal{O})$. For $u \in \mathcal{O}$, let $d_{\mathcal{O}}(u) = |u \sqcap (\mathcal{O} - u)|$, where $u \sqcap (\mathcal{O} - u) = \{x \in \mathcal{O} - u \mid x \cap u \neq \emptyset\}$. Then, for all $u \in \mathcal{O}$, we have $d_{\mathcal{O}}(u) \geq \nabla$,*

Proof. Observe that

$$\nabla = \frac{|\mathsf{E}_{\mathcal{O}}|}{|\mathcal{O}|} = \frac{|\mathsf{E}_{\mathcal{O}-u}| + d_{\mathcal{O}}(u)}{|\mathcal{O}| - 1 + 1}.$$

As such, if $d_{\mathcal{O}}(u) < \nabla$, then

$$\frac{d_{\mathcal{O}}(u)}{1} < \nabla = \frac{d_{\mathcal{O}}(u) + |\mathsf{E}_{\mathcal{O}-u}|}{1 + |\mathcal{O}| - 1} < \frac{|\mathsf{E}_{\mathcal{O}-u}|}{|\mathcal{O}| - 1}.$$

But this implies that $\mathcal{O} - u$ is denser than $\mathcal{O}$, which is a contradiction. ◀

[1] Indeed, $(1 + c_1\varepsilon/c)(1 + c_2\varepsilon/c) \leq 1 + (c_1/c + c_2/c + c_1c_2/c^2)\varepsilon \leq 1 + (c_1 + c_2 + 1)\varepsilon/c$.

2.2 Reporting all intersecting pairs of disks

The algorithm of Section 5 requires an efficient algorithm to report all the intersecting pairs of disks.

▶ **Lemma 7.** *Given a set $\mathcal{D}$ of n disks, all the intersecting pairs of disks of $\mathcal{D}$ can be computed in $O(n \log n + k)$ expected time, where k is the number of intersecting pairs.*

Proof. We break the boundary of each disk at its two x-extreme points, resulting in a set of $2n$ x-monotone curves. Computing the vertical decomposition of the arrangement of these disks (curves) $\mathcal{A}(\mathcal{D})$ can be done in $O(n \log n + k)$ expected time [20]. See Figure 2.1 for an example. This gives us readily all the pairs that their boundaries intersect.

As for the intersections that rise out of containment, perform a traversal of the dual graph of the vertical decomposition (i.e., each vertical trapezoid is a vertex, and two trapezoids are adjacent if they share a boundary edge). The dual graph is planar with $O(n + k)$ vertices and edges, and as such the graph can be traversed in $O(n + k)$ time. During the traversal, by appropriate bookkeeping, it is straightforward to maintain the list of disks containing the current trapezoid, in $O(1)$ per edge traversed, as any edge traversed changes this set by at most one.

For a disk d, let p_{d} be the rightmost point of d. For each disk d, pick any trapezoid Δ such that $p_{\mathsf{d}} \in \Delta$ and either the top boundary of d is the ceiling of Δ or the bottom boundary of d is the floor of Δ. Assign $\Delta_{\mathsf{d}} \leftarrow \Delta$ as the *representative* trapezoid of d.

During the traversal of the dual graph, consider the case where we arrive at a representative trapezoid of a disk d. Let $L(\mathsf{d})$ be the list of disks containing Δ_{d}. Then scan $L(\mathsf{d})$ to report all the containing pairs of d. Each disk in $L(\mathsf{d})$ either intersects the boundary of d, or contain it. Therefore, the total time spent at the representative trapezoids is $\sum_{\mathsf{d} \in \mathcal{D}} |L(d)| = O(k)$. ◀

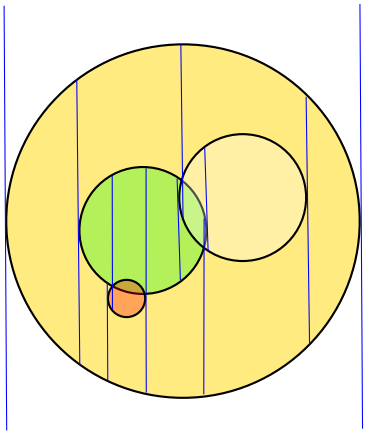

Figure 2.1 Vertical decomposition of four disks.

3 From reporting to approximate sampling/counting

In this section, given a set of objects $\mathcal{D}$, and a reporting data-structure for $\mathcal{D}$, we show a reduction to building a data-structure, such that given a query object q, it returns an object from $\mathsf{q} \sqcap \mathcal{D} = \{x \in \mathcal{D} \mid x \cap \mathsf{q} \neq \emptyset\}$, with almost uniform probability, and also returns an $(1 \pm \varepsilon)$-approximation for the size of this set.

3.1 The data-structure

The given reporting data-structure. Let $\mathcal{D}$ be a set of n objects, and assume for any subset $X \subseteq \mathcal{D}$ of size m, one can construct, in $C(m)$ time, a data-structure that given a query object d, returns, in $O(Q(m) + k)$ time, all the objects in X that intersects d, where $k = |\mathsf{d} \sqcap X|$. Furthermore, we assume that if a parameter k' is specified by the query, then the data-structure stops after $O(Q(m) + k')$ time, if $k > k'$, and indicate that this is the case.

▶ **Example 8.** If $\mathcal{D}$ is a set of disks, and the query d is a disk, then this becomes a query reporting of the k-nearest neighbors in an additive weighted Voronoi diagram. Liu [23] showed how to build such a reporting data-structure, in $O(n \log n)$ expected preprocessing time, and query time $O(\log n + k)$.

Data-structure construction. We build a random binary tree over the objects of $\mathcal{D}$, by assigning each object of $\mathcal{D}$ with equal probability either a 0 or a 1 label. This partitions $\mathcal{D}$ into two sets $\mathcal{D}_0$ (label 0) and $\mathcal{D}_1$ (label 1). Recursively build random trees T_0 and T_1 for $\mathcal{D}_0$ and $\mathcal{D}_1$, respectively, with the output tree having T_0 and T_1 as the two children of the root. The constructions bottoms out when the set of objects is of size one. Let $\mathcal{T}$ be the resulting tree that has exactly n leaves. For every node u of $\mathcal{T}$, we construct the reporting data-structure for the set of objects $\mathcal{D}(u)$ – that is, the set of objects stored in the subtree of u.

Finally, create an array L_i for each level i of the tree $\mathcal{T}$, containing pointers to all the nodes in this level of the tree.

Answering a query. Given a query object q, and a parameter $\varepsilon \in (0, 1/2)$, the algorithm starts from an arbitrary leaf v of $\mathcal{T}$. The leaf v has a unique root to v path, denoted by $\pi = u_0 u_1 \ldots u_t$, where $u_0 = \mathrm{root}(\mathcal{T})$ and $u_t = v$. The algorithm performs a binary search on π, using the reporting data-structure associated with each node, to find the maximal i, such that $|\mathsf{q} \sqcap \mathcal{D}(u_i)| > \psi = c \log n$, where c is a sufficiently large constant. Here, we use the property that one can abort the reporting query if the number of reported objects exceeds ψ. This implies that each query takes $Q(n) + O(\log n)$ time (with no dependency on ε). Next, the algorithm computes the maximal j such that $|\mathsf{q} \sqcap \mathcal{D}(u_j)| > \psi_\varepsilon = c\varepsilon^{-2} \log n$ (or set $j = 0$ if such a j does not exist). This is done by going up the path π from u_i, trying $u_{i-1}, u_{i-2}, \ldots, u_j$ using the reporting data-structure till the condition is fulfilled[2]. Next, the algorithm chooses a vertex $u \in L_j$ uniformly at random. It computes the set $S = \mathsf{q} \sqcap \mathcal{D}(u)$ using the reporting data-structure. The algorithm then returns a random object from S uniformly at random, and the number $2^j |S|$. The first is a random element chosen from $\mathsf{q} \sqcap \mathcal{D}$, and the second quantity returned is an estimate for $|\mathsf{q} \sqcap \mathcal{D}|$.

3.2 Analysis

3.2.1 Correctness

▶ **Lemma 9.** *Let $\varepsilon \in (0, 1/2)$, $\psi_\varepsilon = c\varepsilon^{-2} \log n$, and q be a query object. Let $M \geq 1$ be the integer such that $\psi_\varepsilon/16 \leq |\mathcal{D} \sqcap \mathsf{q}| / 2^M < \psi_\varepsilon/8$. Then, for all nodes v at distance $i \leq M$ from the root of $\mathcal{T}$, we have $\mathbb{P}\left[|\mathcal{D}(v) \sqcap \mathsf{q}| \notin (1 \pm \varepsilon/2) \frac{|\mathcal{D} \sqcap \mathsf{q}|}{2^i}\right] \leq \frac{1}{n^{\Omega(c)}}$.*

[2] One can also "jump" to level $i + \log_2(1/\varepsilon^2) - 2$, and do a local search there for j, but this "improvement" does not effect the performance.

Proof. Consider a node v at a distance i from the root, and let $Y_v = |\mathcal{D}(v) \sqcap \mathsf{q}|$. Clearly, $\mu_v = \mathbb{E}[Y_v] = |\mathcal{D} \sqcap \mathsf{q}| / 2^i$. Since $i \leq M$, we have $\mu_v \geq \psi_\varepsilon/16$. By Chernoff's inequality, we have

$$\mathbb{P}\big[Y_v \notin (1 \pm \varepsilon/2)\mu_v\big] \leq 2\exp\big(-\varepsilon^2\mu_v/3\big) \leq 2\exp\big(-\varepsilon^2\psi_\varepsilon/48\big) \leq 2\exp(-(c/48)\log n) \leq \frac{2}{n^{c/48}}.$$

The number of nodes in $\mathcal{T}$ is $O(n)$, and hence, by the union bound, for all nodes v at distance $i \leq M$ from the root, we have $\mathbb{P}[Y_v \notin (1 \pm \varepsilon/2)\mu_v] \leq 1/n^{\Omega(c)}$. ◀

▶ **Observation 10.** *Lemma 9 implies that for all nodes v at distance $i > M$ from the root of $\mathcal{T}$, we have $\mathbb{P}\big[|\mathcal{D}(v) \sqcap \mathsf{q}| > \psi_\varepsilon\big] \leq 1/n^{\Omega(c)}$. Indeed, Lemma 9 implies this for all nodes at distance M from the root, and the sizes of these sets are monotonically decreasing along any path down the tree.*

▶ **Lemma 11.** *Assume that the number of distinct sets $\mathsf{q}' \sqcap \mathcal{D}$, over all possible query objects q', is bounded by a polynomial $O(n^d)$, where d is some constant. Then, for a query q, the probability that the algorithm returns a specific object $\mathsf{o} \in \mathcal{D} \sqcap \mathsf{q}$, is in $(1 \pm \varepsilon)/\beta$, where $\beta = |\mathcal{D} \sqcap \mathsf{q}|$. Similarly, the estimate the algorithm outputs for β is in $(1 \pm \varepsilon)\beta$. The answer is correct for all queries, with probability $\geq 1 - 1/n^{\Omega(c)}$, for a sufficiently large constant c.*

Proof. It is easy to verify the algorithm works correctly if $|\mathsf{q} \sqcap \mathcal{D}(u_j)| < \psi_\varepsilon$. Otherwise, for the node u_j computed by the algorithm, we have $|\mathsf{q} \sqcap \mathcal{D}(u_j)| > \psi_\varepsilon = c\varepsilon^{-2}\log n$. By Observation 10, with high probability, we have that $j \leq M$. By Lemma 9, it implies that for any node $u \in L_j$, we have $2^j\,|\mathcal{D}(u) \sqcap \mathsf{q}| \in (1 \pm \varepsilon/2)\,|\mathcal{D} \sqcap \mathsf{q}| = (1 \pm \varepsilon/2)\beta$, which implies that the estimate for the size of β is correct, as $u \in L_j$. This readily implies that the probability of returning a specific object $\mathsf{o} \in \mathcal{D} \sqcap \mathsf{q}$ is in $(1 \pm \varepsilon)/\beta$, since

$$\frac{1-\varepsilon}{\beta} \leq \frac{1}{(1+\varepsilon/2)\,|\mathcal{D} \sqcap \mathsf{q}|} \leq \frac{1}{|L_j| \cdot |\mathcal{D}(u) \sqcap \mathsf{q}|} \leq \frac{1}{(1-\varepsilon/2)\,|\mathcal{D} \sqcap \mathsf{q}|} \leq \frac{1+\varepsilon}{\beta}.$$

As for the probabilities, there are n nodes in $\mathcal{T}$, and $O(n^d)$ different queries, and thus the probability of failure is at most $n^{d+1}/n^{\Omega(c)} < 1/n^{\Omega(c)}$, by Lemma 9. ◀

3.2.2 Running times

Query time. The depth of $\mathcal{T}$ is $h = O(\log n)$ with high probability (follows readily from Chernoff's inequality). Thus, the first stage (of computing the maximal i) requires $O(\log\log n)$ queries on the reporting data-structure, where each query takes $O(Q(n) + \log n)$ time. The second stage (of finding maximal j) takes

$$\tau = \mathbb{E}\Big[\sum\nolimits_{t=j}^{i} O(Q(n) + |\mathcal{D}(u_t) \sqcap \mathsf{q}|)\Big].$$

Thus, we have

(A) If $Q(n) = O(\log n)$, then $\tau = O(\psi_\varepsilon)$, as the cardinality of $\mathcal{D}(u_t) \sqcap \mathsf{q}$ decreases by a factor of two (in expectation) as one move downward along a path in the tree. Thus τ is a geometric summation dominated by the largest term.

(B) If $Q(n) = \Omega(\log n)$, we have (in expectation) that $|i - j| \leq O(\log(1/\varepsilon))$, and thus $\tau = O(Q(n)\log(1/\varepsilon) + \psi_\varepsilon)$ time.

(C) If $Q(n) = O(n^\lambda)$, for $0 < \lambda \leq 1$, then the query time is dominated by the query time for the top node (i.e., u_j) in this path, and $\tau = O(Q(n))$, as can easily be verified.

Construction time. The running time bounds of the form $O(C(n))$ are *well-behaved*, if for any non-negative integers $n_1, n_2, \ldots$, such that $\sum_{i=1} n_i = n$, implies that $\sum_{i=1} C(n_i) = O(C(n))$. Under this assumption on the construction time, we have that the total construction time is $O(C(n) \log n)$.

3.2.3 Summary

▶ **Theorem 12.** *Let $\mathcal{D}$ be a set of n objects, and assume we are given a well-behaved range-reporting data-structure that can be constructed in $C(m)$ time, for m objects, and answers a reporting query q in $O(Q(m) + |\mathsf{q} \sqcap \mathcal{D}|)$ time. Then, one can construct a data-structure, in $O(C(n) \log n)$ time, such that given a query object q, it reports an $(1 \pm \varepsilon)$-estimate for $\beta = |\mathsf{q} \sqcap \mathcal{D}|$, and also returns an object from $\mathsf{q} \sqcap \mathcal{D}$, where each object is reported with probability $(1 \pm \varepsilon)/\beta$. The data-structure answers all such queries correctly with probability $\geq 1 - 1/n^{\Omega(1)}$. The expected query time is:*

- **(i)** $O((\varepsilon^{-2} + \log\log n) \log n)$ *if* $Q(m) = O(\log m)$.
- **(ii)** $O(Q(n))$ *if* $Q(m) = O(m^{\lambda})$, *for some constant* $\lambda > 0$.
- **(iii)** $O\big(\varepsilon^{-2} \log n + Q(n) \log \frac{\log n}{\varepsilon}\big)$ *otherwise.*

Plugging in the data-structure of Liu [23] for disks, with $C(m) = O(m \log m)$, and $Q(m) = O(\log m)$, in the above theorem, implies the following.

▶ **Corollary 13.** *Let $\mathcal{D}$ be a set of n disks in the plane. One can construct in $O(n \log^2 n)$ time a data-structure, such that given a query disk q and a parameter $\varepsilon \in (0, 1/2)$, it outputs an $(1 \pm \varepsilon)$-estimate for $\beta = |\mathsf{q} \sqcap \mathcal{D}|$, and also returns a disk in $\mathsf{q} \sqcap \mathcal{D}$ with a probability that is $(1 \pm \varepsilon)$-uniform. The expected query time is $O((\varepsilon^{-2} + \log\log n) \log n)$, and the result returned is correct with high probability for all possible queries.*

Approximate depth queries for a fixed threshold. If we are interested only in a single threshold, the above data-structure is an overkill, as one can directly construct a single sample and perform the query directly on this sample. This implies the following.

▶ **Corollary 14.** *Let $\mathcal{D}$ be a set of n disks in the plane. One can construct, in $O(n \log n)$ time, a data-structure, such that given a query disk q, and a parameter $\varepsilon \in (0, 1/2)$, it reports whether $|\mathsf{q} \sqcap \mathcal{D}| \geq \beta$ or $|\mathsf{q} \sqcap \mathcal{D}| < \beta$. The data structure is allowed to make a mistake when $|\mathsf{q} \sqcap \mathcal{D}| \in [(1-\varepsilon)\beta, (1+\varepsilon)\beta]$. The data-structure answers the query correctly with probability $\geq 1 - 1/n^{\Omega(1)}$, and the query time is $O(\varepsilon^{-2} \log n)$.*

4 A $(2+\varepsilon)$-approximation for densest subset disks

In this section, we design a $(2+\varepsilon)$-approximation algorithm to compute the densest subset of disks in $O(\varepsilon^{-4} n \log n)$ time.

4.1 Constant approximation via depth

The ***depth*** of a point in the arrangement of disks is the number of disks containing the point. The ***deepest*** point in the plane is the point with the maximum depth. The deepest point in the arrangement of disks and the densest subgraph in the geometric intersection graph of disks are the same (up to a constant factor).

▶ **Lemma 15.** *Let $\hbar$ be the depth of the deepest point in $\mathcal{D}$, where $\mathcal{D}$ is a set of disks in the plane, and let $\mathscr{d}$ be maximum density of $\mathcal{D}$. Then $(\hbar - 1)/2 \leq \mathscr{d} \leq 7 \cdot \hbar$, where c is a sufficiently large constant. In particular, in $O(n \log n)$ time, one can compute a quantity t, such that $(t-1)/2 \leq \mathscr{d} \leq 8t$.*

Proof. Lemma 6 readily implies that $\mathscr{d} \geq (\hbar - 1)/2$, as this is the density provided by the disks containing the deepest point.

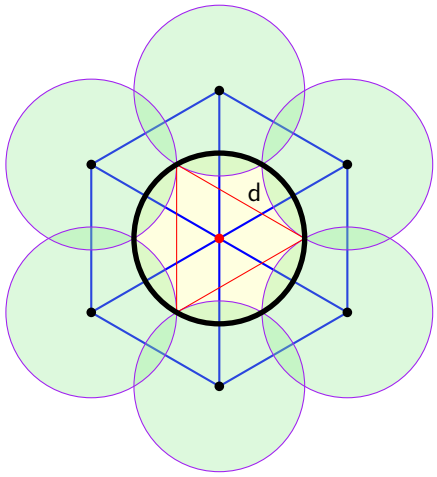

Figure 4.1 Defending a disk by a guard set made of 7 points.

As for the other direction, consider the densest subset $\mathcal{O} \subseteq \mathcal{D}$, and let d be the smallest disk in S. By Lemma 6, the set $T = \mathsf{d} \sqcap (\mathcal{O} - \mathsf{d})$ has size least $\mathscr{d}$. Let c be the center of d and inscribe an equilateral triangle in d. Consider a tiling of 6 such triangles that are interior disjoint, all sharing a vertex at c. Let P be the set of 7 vertices used by these triangles (including c), see Figure 4.1. One can verify[3] that any disk d' at least as large as d that intersect d, must contain at least one point of P.

Thus, each disk in T is stabbed by at least one point in P, which readily implies that there is a point in P that has depth $|T| / |P| \geq \mathscr{d}/7$.

As for the last part, a 1.1-approximation of the deepest point in the arrangement of the disks can be computed in $O(n \log n)$ time [5]. The returned approximate deepest point provides the desired approximation. ◀

4.2 The algorithm

The input is a set $\mathcal{D}$ of n disks in the plane. Let $\vartheta = \varepsilon/15$. The algorithm starts by computing a number ξ, such that $\mathscr{d} \in [(\xi - 1)/2, 8\xi]$, using the algorithm of Lemma 15. The basic idea is to try a sequence of exponentially decaying values to the optimal density $\mathscr{d}$. To this end, in the ith round, the algorithm tries the degree threshold $\beta = 8\xi(1-\vartheta)^i$, for $i = 1, \ldots, \log_{1+\vartheta} 16 = O(1/\varepsilon)$.

In the beginning of such a round, let $\mathcal{L} \leftarrow \mathcal{D}$. During a round, the algorithm repeatedly removes "low-degree" objects, by repeatedly doing the following:

[3] Indeed, if the center c' of d' is in the union of the seven copies of d centered at the points of P, see Figure 4.1, then the disk d' must contain one of the points of P. Otherwise, it can be verified that d and d' do not intersect.

(I) Construct the data-structure of Corollary 14 on the objects of $\mathcal{L}$.

(II) Let $\mathcal{L}_< \subseteq \mathcal{L}$ be the objects whose degree in $\mathcal{L}$ is smaller than $(1+\vartheta)\beta$ according to this data-structure (i.e., query all the objects of $\mathcal{L}$ for their degree). Let $\mathcal{L}_\geq = \mathcal{L} \setminus \mathcal{L}_<$.

(III) If $\mathcal{L}_\geq$ is empty, then this round failed, and the algorithm continues to the next round.

(IV) If $|\mathcal{L}_<| < \vartheta\,|\mathcal{L}|$, then the algorithm returns $\mathcal{L}$ as the desired approximate densest subset.

(V) Otherwise, let $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{L}_<$. The algorithm continues to the next iteration (still inside the same round).

4.3 Analysis

▶ **Lemma 16.** *When the algorithm terminates, we have $\beta \geq (1-\vartheta)^3 \boldsymbol{d}$, with high probability, where $\boldsymbol{d}$ is the optimal density.*

Proof. Consider the iteration when $\beta \in [(1-\vartheta)^3\boldsymbol{d}, (1-\vartheta)^2\boldsymbol{d}]$. By definition of $\mathcal{L}_<$, all the objects in it have a degree at most $(1+\vartheta)\beta \leq (1+\vartheta)(1-\vartheta)^2\boldsymbol{d} \leq (1-\vartheta)\boldsymbol{d}$. By Lemma 6, all the objects in the optimal solution have degree $\geq \boldsymbol{d}$ (when restricted to the optimal solution). Therefore, none of the objects in the optimal solution are in $\mathcal{L}_<$ and hence, the set $\mathcal{L}_\geq$ is not empty (and contains the optimal solution). Inside this round, the loop is performed at most $O(\vartheta^{-1}\log n)$ times, as every iteration of the loop shrinks $\mathcal{L}$ by a factor of $1-\vartheta$. This implies that the algorithm must stop in this round. ◀

▶ **Lemma 17.** *The above algorithm returns a $(2+\varepsilon)$-approximation of the densest subset.*

Proof. Consider the set $\mathcal{L}$, the value of β when the algorithm terminated, and let $\nu = |\mathcal{L}|$. By Lemma 16, $\beta \geq (1-\vartheta)^3\boldsymbol{d}$, and by the algorithm stopping condition, we have $|\mathcal{L}_<| < \vartheta\,|\mathcal{L}|$. In addition, all the objects in $\mathcal{L}_\geq$ have degree at least $(1-\vartheta)\beta$. Thus, the number of induced edges on $\mathcal{L}$ is

$$\mathsf{m}(\mathcal{L}) \geq \frac{(1-\vartheta)\beta\,|\mathcal{L}_\geq|}{2} > \frac{(1-\vartheta)^2\beta\,|\mathcal{L}|}{2} \geq (1-\vartheta)^5\frac{\boldsymbol{d}}{2}\,|\mathcal{L}| \geq (1-5\vartheta)\frac{\boldsymbol{d}}{2}\,|\mathcal{L}|\,.$$

Thus $\nabla(\mathcal{L}) = \frac{\mathsf{m}(\mathcal{L})}{|\mathcal{L}|} \geq (1-5\vartheta)\boldsymbol{d}/2 = (1-\varepsilon/3)\boldsymbol{d}/2$, as $\vartheta = \varepsilon/15$. Observe that $2/(1-\varepsilon/3) \leq 2(1+\varepsilon/2) \leq 2+\varepsilon$. ◀

▶ **Lemma 18.** *The expected running time of the above algorithm is $O(n\varepsilon^{-4}\log n)$.*

Proof. The expected time spent, in an iteration, in step I and II is $O(\vartheta^{-2}n\log n)$, where $n = |\mathcal{L}|$, and this dominates the running time of this iteration. Let n_i be the size of $\mathcal{L}$ in the ith iteration (inside the same round), and observe that $n_{i+1} \leq (1-\vartheta)n_i$. Assume t iterations are performed inside this round. As $\sum_{i=1}^{t} n_i = O(n/\vartheta)$, the expected running time for the round is $\sum_{i=1}^{t} O(\vartheta^{-2}n_i\log n_i) = O(\vartheta^{-3}n\log n)$. The value of β can change $O(1/\varepsilon)$ times during the algorithm and hence, the overall expected running time of the algorithm is $O(n\vartheta^{-4}\log^2 n)$, which implies the result since $\vartheta = \Theta(\varepsilon)$. ◀

Summarizing, we get the following.

▶ **Theorem 19.** *Let $\mathcal{D}$ be a set of n disks in the plane, and let $\varepsilon \in (0, 1/2)$ be a parameter. The overall algorithm computes, in $O(\varepsilon^{-4}n\log n)$ expected time, a $(2+\varepsilon)$-approximation to the densest subgraph of $\mathsf{G}_{\cap\mathcal{D}}$, and the result is correct with high probability.*

▶ Remark 20 (A lower bound). Consider the element uniqueness problem – of deciding if n real numbers are all distinct. It is known that in the comparison model this requires $\Omega(n \log n)$ time. Treating each number as a disk of radius zero, readily implies there is a subgraph of density ≥ 1 (and in particular, non-zero), if and only if there are two identical numbers. Thus, any approximation algorithm for the maximum density problem requires $\Omega(n \log n)$ time (in the comparison model).

5 An $(1+\varepsilon)$-approximation for densest subset disks

Here, we present a $(1+\varepsilon)$-approximation algorithm for the densest subset of disks, which is based on the following intuitive idea – if the intersection graph is sparse, then the problem is readily solvable. If not, then one can sample a sparse subgraph, and use an approximation algorithm on the sampled graph.

5.1 Densest subgraph estimation via sampling

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a graph with n vertices and m edges, with maximum subgraph density $\mathbb{d}$. Let $\vartheta \in (0, 1/6)$ be a parameter, and assume that $m > c' n \vartheta^{-2} \log n$, where c' is some sufficiently large constant, which in particular implies that

$$\mathbb{d} = \mathbb{d}(\mathsf{G}) \geq \frac{m}{n} \geq \frac{c'}{\vartheta^2} \log n.$$

Assume we have an estimate $\overline{m} \in (1 \pm \vartheta)m$ of m. For a constant c to be specified shortly, with $c < c'$, let

$$\psi = c \frac{n}{\overline{m}} \vartheta^{-2} \log n \leq \frac{c}{c'(1-\vartheta)} \leq \frac{6c}{5c'} < 1.$$

Let $F = \{e_1, \ldots, e_r\}$ be a random sample of $r = \lceil \psi \overline{m} \rceil$ edges from G. Specifically, in the ith iteration, an edge e_i is picked from the graph, where the probability of picking any edge is in $(1 \pm \vartheta)/m$. Let $H = (V, F)$, and observe that H is a sparse graph with n vertices and $r = O(\vartheta^{-2} n \log n)$ edges. The claim is that the densest subset $D \subseteq V$ in H, or even approximate densest subset, is close to being the densest subset in G. The proof of this follows from previous work [24], but requires some modifications, since we only have an estimate to the number of edges m, and we are also interested in approximating the densest subgraph on the resulting graph. We include the details here so that the presentation is self contained. The result we get is summarized in Lemma 24, if the reader is uninterested in the (somewhat tedious) analysis of this algorithm.

5.1.1 Analysis

▶ **Lemma 21.** *Let $\mathbb{d} = \mathbb{d}(\mathsf{G})$, and let $U \subseteq V$, be an arbitrary set of k vertices. If $\nabla_{\mathsf{G}}(U) \leq \mathbb{d}/60$, then $\mathbb{P}[\nabla_H(U) \geq \psi \mathbb{d}/5] \leq n^{-100k}$.*

Proof. We have $\mathbb{d} \geq m/n$, where $n = |\mathsf{V}(\mathsf{G})|$ and $m = |\mathsf{E}(\mathsf{G})|$, and thus

$$\psi = c \frac{n}{\overline{m}\vartheta^2} \log n \geq c \frac{n}{(1+\vartheta)m\vartheta^2} \log n \geq \frac{1}{\mathbb{d}} \cdot \frac{c}{(1+\vartheta)\vartheta^2} \log n$$

Let $X_i = 1$ if the edge sampled in the ith round belongs to H_U and zero, otherwise. Let $X = \sum_i X_i$ be the number of edges in H_U. Then

$$\mathbb{P}[X_i = 1] \in (1 \pm \vartheta) \frac{|\mathsf{E}(\mathsf{G}_U)|}{m} = (1 \pm \vartheta) \frac{k \nabla_{\mathsf{G}}(U)}{m}.$$

By linearity of expectations, and as $\overline{m} \in (1 \pm \vartheta)m$, we have

$$\mathbb{E}[X] \in (1 \pm \vartheta)\psi\overline{m}\frac{k\nabla_{\mathsf{G}}(U)}{m} \subseteq (1 \pm \vartheta)^2\psi k\nabla_{\mathsf{G}}(U). \tag{5.1}$$

By assumption $\nabla_{\mathsf{G}}(U) \leq \mathbf{d}/60$, implying that $\mathbb{E}[X] \leq (1 + 3\vartheta)\psi k\mathbf{d}/60 \leq \psi k\mathbf{d}/30$, if $\vartheta \in (0, 1/3)$, by Observation 1. Observe that $(1 + (2e - 1))\,\mathbb{E}[X] \leq \frac{\psi k d}{5}$. By Chernoff's inequality, Lemma 30, we have

$$\mathbb{P}\left[\nabla_H(U) \geq \frac{\psi\mathbf{d}}{5}\right] = \mathbb{P}\left[X \geq \frac{\psi k\mathbf{d}}{5}\right] \leq 2^{-\psi k d/5} \leq \frac{1}{n^{100k}},$$

by picking c to be sufficiently large. ◀

▶ **Lemma 22.** *Let $\mathbf{d} = \mathbf{d}(\mathsf{G})$, and let $U \subseteq V$, be an arbitrary set of k vertices. If $\nabla_{\mathsf{G}}(U) \geq \mathbf{d}/60$, then $\mathbb{P}\big[\nabla_H(U) \in (1 \pm \vartheta)^3\psi\nabla_{\mathsf{G}}(U)\big] \geq 1 - n^{-100k}$.*

Proof. Following the argument of Lemma 21 and as $\nabla_{\mathsf{G}}(U) \geq \mathbf{d}/60$, we have that $\mathbb{E}[X] = \Omega(k \cdot \psi\nabla_{\mathsf{G}}(U)) = \Omega(k \cdot \psi\mathbf{d}) = \Omega(k \cdot c\vartheta^{-2}\log n)$. Chernoff's inequality, Theorem 31, then implies that $X \in (1 \pm \vartheta)\,\mathbb{E}[X]$ with probability at least $1 - 2\exp(-\vartheta^2\,\mathbb{E}[X]\,/4) \geq 1 - 1/n^{100k}$, for n sufficiently large. The claim now readily follows from Eq. (5.1). ◀

▶ **Lemma 23.** *Let $\alpha \in (0, 1/6)$ be a parameter. For all sets $U \subseteq V$, such that $\nabla_H(U) \geq (1 - \alpha)\mathbf{d}(H)$, we have that $\nabla_{\mathsf{G}}(U) \geq (1 - 6\vartheta)(1 - \alpha)\mathbf{d}$, and this holds with high probability.*

Proof. Let X be the densest subset in G. By Lemma 22, we have that

$$\nabla_H(X) \in (1 \pm \vartheta)^3\psi\mathbf{d}(\mathsf{G}) \implies \mathbf{d}(H) \geq (1 - \vartheta)^3\psi\mathbf{d} \geq \frac{\psi\mathbf{d}}{2}.$$

By Lemma 21, we have that for all the sets $T \subseteq \mathsf{V}$, with $\nabla_{\mathsf{G}}(T) \leq \mathbf{d}/60$, we have $\nabla_H(T) < \psi\mathbf{d}/5 < \mathbf{d}(H)/2$, and this happens with probability $\sum_{k=2}^{n}\sum_{T\subseteq\mathsf{V}:|T|=k} 1/n^{100k} \leq \sum_{k=2}^{n}\binom{n}{k}/n^{100k} \leq 1/n^{99}$.

Thus, all the sets $U \subseteq \mathsf{V}$ under consideration have $\nabla_{\mathsf{G}}(U) > \mathbf{d}/60$. By Lemma 22, for all such sets, with probability $1 - n^{-100k} \geq 1 - 1/n^{99}$, we have $\nabla_H(U) \in (1 \pm \vartheta)^3\psi\nabla_{\mathsf{G}}(U)$, which implies $\nabla_{\mathsf{G}}(U) \in \frac{1}{(1\pm\vartheta)^3\psi}\nabla_H(U)$. Thus, we have

$$\nabla_{\mathsf{G}}(U) \geq \frac{1}{(1+\vartheta)^3\psi}\nabla_H(U) \geq \frac{(1-\alpha)\mathbf{d}(H)}{(1+\vartheta)^3\psi} \geq \frac{(1-\alpha)(1-\vartheta)^3\psi\mathbf{d}}{(1+\vartheta)^3\psi} \geq (1-\alpha)(1-6\vartheta)\mathbf{d},$$

since $1/(1 + \vartheta) \geq 1 - \vartheta$, and $(1 - \vartheta)^6 \geq 1 - 6\vartheta$. ◀

5.1.2 Summary

▶ **Lemma 24.** *Let $\varepsilon \in (0, 1)$ be a parameter, and let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a graph with n vertices and m edges, with $m = \Omega(\varepsilon^{-2}n\log n)$. Furthermore, let $\overline{m}$ be an estimate to m, such that $\overline{m} \in (1 \pm \vartheta)m$, where $\vartheta = \varepsilon/10$. Let $\psi = c(n/\overline{m})\vartheta^{-2}\log n$, and let F be a random sample of $\psi\overline{m} = O(\varepsilon^{-2}n\log n)$ edges, with repetition, where the probability of any specific edge to be picked is $(1 \pm \vartheta)/m$, and c is a sufficiently large constant. Let $H = (\mathsf{V}, F)$ be the resulting graph, and let $X \subseteq \mathsf{V}$ be subset of H with $\nabla_H(X) \geq (1 - \varepsilon/6)\mathbf{d}(H)$. Then, $\nabla(X) \geq (1 - \varepsilon)\mathbf{d}(\mathsf{G})$.*

Proof. This follows readily from the above, by setting $\alpha = \varepsilon/6$, and using Lemma 23. ◀

5.2 Random sampler

To implement the above algorithm, we need an efficient algorithm for sampling edges from the intersection graph of disks, which we describe next.

5.2.1 The algorithm

The algorithm consists of the following steps:

(I) Build the data-structure of Corollary 13 on the disks of $\mathcal{D}$ with error parameter ε/c, where c is a sufficiently large constant. Also, build the range-reporting data-structure of Liu [23] on the disks of $\mathcal{D}$.

(II) For each object $\mathsf{o} \in \mathcal{D}$, query the data-structure of Corollary 13 with o. Let the estimate returned be d'. If $d' < c'/\varepsilon$ (for a constant $c' \gg c$), then report $\mathsf{o} \sqcap \mathcal{D}$ by querying the range-reporting data-structure with o, and set $d_{\mathsf{o}} \leftarrow |\mathsf{o} \sqcap \mathcal{D}| - 1$. Otherwise, set $d_{\mathsf{o}} \leftarrow d'$.

(III) We perform $|F|$ iterations and in each iteration, sample a random edge from $\mathsf{G}_{\cap\mathcal{D}}$. In a given iteration, sample a disk $\mathsf{o} \in \mathcal{D}$, where o has a probability of $\frac{d_{\mathsf{o}}}{\sum_{\mathsf{o}\in\mathcal{D}} d_{\mathsf{o}}}$ being sampled. If $d_{\mathsf{o}} < c'/\varepsilon$, then uniformly-at-random report a disk from $\mathsf{o} \sqcap (\mathcal{D} - \mathsf{o})$. Otherwise, query the data-structure of Corollary 13 with o which returns a disk in $\mathsf{o} \sqcap \mathcal{D}$ (keep querying till a disk other than o is returned).

5.2.2 Analysis

▶ **Lemma 25.** *For each object $\mathsf{o} \in \mathcal{D}$, we have $d_{\mathsf{o}} \in (1 \pm \varepsilon/c'')|\mathsf{o} \sqcap (\mathcal{D} - \mathsf{o})|$, where $c \gg c''$ with high probability.*

Proof. Fix an object o. When $d' < c'/\varepsilon$, then the statement holds trivially. Let $d = |\mathsf{o} \sqcap (\mathcal{D} - \mathsf{o})|$. Now we consider the case $d' \geq c'/\varepsilon$. We know that $d' \in (1 \pm \varepsilon/c)(d+1)$. Firstly, $d' \leq (1+\varepsilon/c)(d+1) \leq (1+\varepsilon/c'')d$ hold if

$$d \geq \frac{1}{\varepsilon} \cdot \frac{1+\varepsilon/c}{1/c'' - 1/c} \geq 1/2\varepsilon.$$

Observe that $d \geq 1/2\varepsilon$, since $c'/\varepsilon \leq d' \leq (1+\varepsilon/c)(d+1)$ implies that $d \geq \frac{c'}{\varepsilon(1+\varepsilon/c)} - 1 \geq 1/2\varepsilon$. Finally, $d' \geq (1-\varepsilon/c)(d+1) \geq (1-\varepsilon/c'')d$ holds trivially. Therefore,

$$d_{\mathsf{o}} = d' \in (1 \pm \varepsilon/c'')d. \qquad \blacktriangleleft$$

▶ **Lemma 26.** *In each iteration, the probability of sampling any edge in $\mathsf{G}_{\cap\mathcal{D}}$ is $(1 \pm \varepsilon)/m$.*

Proof. An edge (u, v) in $\mathsf{G}_{\cap\mathcal{D}}$ can get sampled in step (III) in two ways. In the first way, the disk corresponding to u gets sampled and then v gets reported as the random neighbor of u, and vice-versa for the second way.

Let $d = |\mathsf{o} \sqcap (\mathcal{D} - \mathsf{o})|$, where o is the disk corresponding to u. Consider the case, where $d_{\mathsf{o}} \geq c'/\varepsilon$. As such, the first way of sampling the edge (u, v) has probability lower-bounded by:

$$\frac{d_{\mathsf{o}}}{\sum d_{\mathsf{o}}} \cdot \underbrace{\frac{1 \pm \varepsilon/c}{d}}_{\text{almost-uniformity}} \subseteq \underbrace{\frac{d_{\mathsf{o}}}{(1 \pm \varepsilon/c'')2m}}_{\text{Lemma 25}} \cdot \frac{1 \pm \varepsilon/c}{d} \subseteq \frac{(1 \pm \varepsilon/c'')d}{(1 \pm \varepsilon/c'')2m} \cdot \frac{1 \pm \varepsilon/c}{d} \subseteq \frac{1 \pm \varepsilon}{2m}.$$

Therefore, for the case $d_{\mathsf{o}} \geq c'/\varepsilon$, the probability of sampling the edge (u, v) is $(1 \pm \varepsilon)/m$. Similarly, the statement holds for the case $d_{\mathsf{o}} < c'/\varepsilon$. $\blacktriangleleft$

▶ **Lemma 27.** *The expected running time of the algorithm is* $O(\varepsilon^{-4}n\log^2 n + \varepsilon^{-2}n\log^2 n \cdot \log\log n)$.

Proof. The first step of the algorithm takes $O(n\log^2 n)$ expected time. The second step of the algorithm takes $O(n(\varepsilon^{-2} + \log\log n)\log n)$ expected time. In the third step of the algorithm, each iteration requires querying the data-structure of Corollary 13 $O(1)$ times in expectation. Therefore, the third step takes $O(|F|) \cdot O((\varepsilon^{-2} + \log\log n)\log n) = O(\varepsilon^{-4}n\log^2 n + \varepsilon^{-2}n\log^2 n \cdot \log\log n)$ expected time. ◀

The above implies the following result.

▶ **Lemma 28.** *Let* $\mathcal{D}$ *be a collection of* n *disks in the plane, and let* $\mathsf{G}_{\cap\mathcal{D}}$ *be the corresponding geometric intersection graph with* m *edges. Let* F *be a random sample of* $O(\varepsilon^{-2}n\log n)$ *edges from* $\mathsf{G}_{\cap\mathcal{D}}$*, with repetition, where the probability of any specific edge to be picked is* $(1\pm\varepsilon)/m$*. The edges are all chosen independently into* F*. Then, the algorithm described above, for computing* F*, runs in* $O(\varepsilon^{-4}n\log^2 n + \varepsilon^{-2}n\log^2 n \cdot \log\log n)$ *expected time.*

5.3 The result

▶ **Theorem 29.** *Let* $\mathcal{D}$ *be a collection of* n *disks in the plane. A* $(1+\varepsilon)$*-approximation to the densest subgraph of* $\mathcal{D}$ *can be computed in* $O(\varepsilon^{-4}n\log^2 n + \varepsilon^{-2}n\log^2 n \cdot \log\log n)$ *expected time. The correctness of the algorithm holds with high probability.*

Proof. The case of intersection graph having $O(\varepsilon^{-2}n\log n)$ edges can be handled directly by computing the whole intersection graph in $O(\varepsilon^{-2}n\log n)$ expected time (using Lemma 7).

To handle the other case, we use Lemma 28 to generate a graph $H = (\mathsf{V}, F)$ in $O(\varepsilon^{-4}n\log^2 n + \varepsilon^{-2}n\log^2 n \cdot \log\log n)$ expected time. Since the intersection graph has $\Omega(\varepsilon^{-2}n\log n)$ edges, using Lemma 24, it suffices to compute a $(1-\varepsilon/6)$ approximate densest subgraph of H, which can be computed by the algorithm of [6] in $O(\varepsilon^{-4}n\log^2 n)$ expected time. ◀

6 Extension to other geometric intersection graphs

Although the focus of the paper has been on disks, the techniques described in this paper can be extended to other geometric intersection graphs as well. For any geometric intersection graph with a well-behaved range-reporting data-structure that can be constructed in $C(n)$ time and query time $O(Q(n) + |\mathsf{q} \sqcap \mathcal{D}|)$, the technique in Section 5 gives a $(1+\varepsilon)$-approximation algorithm with a running time of $\tilde{O}(C(n) + nQ(n))$ (where the $\tilde{O}$ notation hides poly-logarithmic factors in n). Therefore, if $C(n) = O(n^{2-\lambda_1})$ and $Q(n) = O(n^{1-\lambda_2})$, it leads to a $(1+\varepsilon)$-approximation algorithm with running time $O(n^{2-\lambda_3})$, where $\lambda_1, \lambda_2, \lambda_3 \in (0,1)$. For example, data structures with such bounds exist for axis-aligned boxes in d-dimension (with $n\log^{O(d)} n$ running time) and spheres in d-dimensions (via reduction to halfspace range reporting). This improves upon the running time of $\Omega(m) = \Omega(n^2)$ obtained for general graphs where the edges are given explicitly.

The technique in Section 4 for disks performs $O(1/\varepsilon)$ rounds by establishing a connection between the deepest point in the arrangement of disks with the maximum density. This connection extends to spheres in d-dimensions but does not hold, for example, for axis-aligned boxes in d-dimensions (in 2-d the depth can be at most two but the maximum density can be $\Omega(n)$). In any case, the technique in Section 4 will perform at most $O(\varepsilon^{-1}\log n)$ rounds if we start guessing the degree threshold from n. Then the running time of the algorithm will be $\tilde{O}(C(n) + nQ(n))$, where $C(n)$ and $Q(n)$ are the construction time and the query

time, respectively, of the approximate depth data-structure for a fixed threshold (analogous version of Corollary 14). Once again this leads to $n \log^{O(d)} n$ running time for axis-aligned boxes in d-dimensions and truly sub-quadratic running time for spheres in d-dimensions.

7 Conclusions

We presented two near-linear time approximation algorithms to compute the densest subgraph on (implicit) geometric intersection graph of disks. We conclude with a few open problems. Are there implicit geometric intersection graphs, such as unit-disk graphs or say, interval graphs, for which the *exact* densest subgraph can be computed in *sub-quadratic time* (in terms of n)? Finally, maintaining the approximate densest subgraph in sub-linear time (again in terms of n) under insertions and deletions of objects, looks to be a challenging problem (in prior work on general graphs an edge gets deleted or inserted, but in an intersection graph a vertex gets deleted or inserted).

References

1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In Claire Mathieu, editor, *Proc. 20th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 180–186. SIAM, 2009. doi:10.1137/1.9781611973068.21.

2 Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Comput. Geom.*, 43(8):700–712, 2010. doi:10.1016/J.COMGEO.2010.04.003.

3 Peyman Afshani and Jeff M. Phillips. Independent range sampling, revisited again. In *Proceedings of Symposium on Computational Geometry (SoCG)*, volume 129, pages 4:1–4:13, 2019. doi:10.4230/LIPICS.SOCG.2019.4.

4 Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proceedings of Symposium on Computational Geometry (SoCG)*, page 271, 2014. doi:10.1145/2582112.2582152.

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

6 Bahman Bahmani, Ashish Goel, and Kamesh Munagala. Efficient primal-dual graph algorithms for mapreduce. In *Algorithms and Models for the Web Graph: 11th International Workshop*, pages 59–78, 2014. doi:10.1007/978-3-319-13123-8_6.

7 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment*, 5(5):454–465, 2012. doi:10.14778/2140436.2140442.

8 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 173–182, 2015. doi:10.1145/2746539.2746592.

9 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discrete & Computational Geometry*, 50(3):771–783, 2013. doi:10.1007/S00454-013-9538-5.

10 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 84–95, 2000. doi:10.1007/3-540-44436-X_10.

11 Chandra Chekuri, Sariel Har-Peled, and Kent Quanrud. Fast LP-based approximations for geometric packing and covering problems. In *Proc. 31st ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 1019–1038, 2020. doi:10.1137/1.9781611975994.62.

12 Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(7):1216–1230, 2012. doi:10.1109/TKDE.2010.271.

13 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

14 Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. *SIAM Journal of Computing*, 32(5):1338–1355, 2003. doi:10.1137/S0097539702403098.

15 Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. Efficient densest subgraph computation in evolving graphs. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 300–310, 2015. doi:10.1145/2736277.2741638.

16 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. Brief announcement: Applications of uniform sampling: Densest subgraph and beyond. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 397–399, 2016. doi:10.1145/2935764.2935813.

17 Giorgio Gallo, Michael D. Grigoriadis, and Robert Endre Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal of Computing*, 18(1):30–55, 1989. doi:10.1137/0218003.

18 David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of Very Large Data Bases (VLDB)*, pages 721–732, 2005. URL: http://www.vldb.org/archives/website/2005/program/paper/thu/p721-gibson.pdf.

19 A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1984.

20 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011. doi:10.1090/surv/173.

21 Samir Khuller and Barna Saha. On finding dense subgraphs. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 597–608, 2009. doi:10.1007/978-3-642-02927-1_50.

22 Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *CoRR*, abs/2303.14467, 2023. doi:10.48550/arXiv.2303.14467.

23 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.

24 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In *Proceedings of Mathematical Foundations of Computer Science (MFCS)*, pages 472–482, 2015. doi:10.1007/978-3-662-48054-0_39.

25 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 1995. doi:10.1017/CBO9780511814075.

26 Saurabh Sawlani and Junxing Wang. Near-optimal fully dynamic densest subgraph. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 181–193, 2020. doi:10.1145/3357713.3384327.

27 Yufei Tao. Algorithmic techniques for independent query sampling. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 129–138, 2022. doi:10.1145/3517804.3526068.

A Chernoff's inequality

The following are standard forms of Chernoff's inequality, see [25].

▶ **Lemma 30.** *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}\big[X\big] = \sum_i p_i$. For $\delta > 2e - 1$, we have $\mathbb{P}\big[X > (1+\delta)\mu\big] < 2^{-\mu(1+\delta)}$.*

▶ **Theorem 31.** *Let $\varepsilon \in (0,1)$ be a parameter. Let $X_1, \ldots, X_n \in \{0,1\}$ be n independent random variables, let $X = \sum_{i=1}^{n} X_i$, and let $\mu = \mathbb{E}[X]$. We have that*

$$\max\Big(\mathbb{P}\big[X < (1-\varepsilon)\mu\big], \mathbb{P}\big[X > (1+\varepsilon)\mu\big]\Big) \leq \exp\big(-\varepsilon^2\mu/3\big)$$