Approximation Algorithms for Maximum Matchings in Geometric Intersection Graphs

Sariel Har-Peled
Department of Computer Science, University of Illinois,
201 N. Goodwin Avenue, Urbana, IL 61801, USA

Everett Yang
Department of Computer Science, University of Illinois,
201 N. Goodwin Avenue, Urbana, IL 61801, USA

Abstract

We present a $(1-\varepsilon)$-approximation algorithms for maximum cardinality matchings in disk intersection graphs – all with near linear running time. We also present an estimation algorithm that returns $(1 \pm \varepsilon)$-approximation to the size of such matchings – this algorithm runs in linear time for unit disks, and $O(n \log n)$ for general disks (as long as the density is relatively small).

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Matchings, disk intersection graphs, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.47

Related Version *Full Version*: `https://arxiv.org/abs/2201.01849`

Funding *Sariel Har-Peled*: Work on this paper was partially supported by a NSF AF award CCF-1907400.

Acknowledgements The authors thank the anonymous referees for their detailed comments.

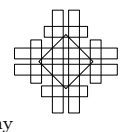
1 Introduction

Geometric intersection graphs

Given a set of n objects U, its intersection graph, $\mathcal{I}_{\mathsf{U}}$, is the graph where the vertices correspond to objects in U and there is an edge between two vertices if their corresponding objects intersect. Such graphs can be dense (i.e., have $\Theta(n^2)$ edges), but they have a linear size representation. It is natural to ask if one can solve problems on such graphs more efficiently than explicitly represented graphs.

Maximum matchings

Computing maximum cardinality matchings is one of the classical problems on graphs (surprisingly, the algorithm to solve the bipartite case goes back to work by Jacobi in the mid 19th century). The fastest combinatorial algorithm (ignoring polylog factors) seems to be the work by Gabow and Tarjan [7], running in $O(m\sqrt{n})$ time where m is the number of edges in the graph. Harvey [11] and Mucha and Sankowski [15] provided algorithms based on algebraic approach that runs in $O(n^\omega)$ time, where $O(n^\omega)$ is the fastest time known for multiplying two $n \times n$ matrices. Currently, the fastest known algorithm for matrix multiplication has $\omega \approx 2.3728596$, but it is far from being practical.



Matchings for planar graphs and disk intersection graphs

Mucha and Sankowski [16] adapted their algebraic technique for planar graphs (specifically using separators), getting running time $O(n^{\omega/2}) \approx O(n^{1.17})$. Yuster and Zwick [17] adapted this algorithm for graphs with excluded minors.

Maximum matchings in geometric intersection graphs

Bonnet et al. [3] studied the problem for geometric intersection graphs. For simplicity of exposition, we describe their results in the context of disk intersection graphs. Given a set n disks with maximum density ρ (i.e., roughly the maximum number of disks covering a point in the plane), they presented an algorithm for computing maximum matchings with running time $O(\rho^{3\omega/2} n^{\omega/2}) \approx O(\rho^{3.5} n^{1.17})$. This compares favorably with the naive algorithm of just plugging such graphs into the algorithm of Gabow and Tarjan, which yields running time $O(m\sqrt{n}) = O(\rho n^{3/2})$. If the ratio between the smallest disk and largest disk is at most Φ, they presented an algorithm with running time $O(\Phi^{12\omega} n^{\omega/2})$. Note that the running time of all these algorithms is super linear in n.

Approximate maximum matchings

It is well known that it is enough to augment along paths of length up to $O(1/\varepsilon)$ if one wants $(1-\varepsilon)$-approximate matchings. For bipartite graphs this implies that one need to run $O(1/\varepsilon)$ rounds of paths finding stage of the bipartite matching algorithm of Hopcroft and Karp [12]. Since such a round takes $O(m)$ time, this readily leads to an $(1-\varepsilon)$-approximate bipartite matching algorithm in this case. The non-bipartite case is significantly more complicated, and the weighted case is even more difficult. Nevertheless, Duan and Pettie [5] presented an algorithm with running time $O(m\varepsilon^{-1} \log \varepsilon^{-1})$ which provides $(1-\varepsilon)$-approximation to the maximum weight matching in non-bipartite graph.

Density and approximate matchings

For a set of objects U in $\mathbb{R}^d$, the density of an object is the number of bigger objects in the set intersecting it. The *density* of the set of objects is the maximum density of the objects. The density is denoted by ρ, and the premise is that for real world inputs it would be small. The intersection graph of such objects when ρ is a constant are known as *low density graphs*, have some nice properties, such as having separators. See [9] and references therein. In particular, for a set of fat objects, the density and the maximum depth (i.e., the maximum number of object covering any point) are roughly the same. It is well known that low density graphs are sparse and have $O(\rho n)$ edges, where n is the number of objects in the set. Since one can compute the intersection graph in $O(n \log n + \rho n)$ time, and plug it into the algorithm of Duan and Pettie [5], it follows that an approximation algorithm with running time $O\big(n \log n + (\rho n/\varepsilon) \log(1/\varepsilon)\big)$. See Section 4.2 for details. Thus, the challenge is to get better running times than this baseline.

1.1 Our results

Our purpose here is to develop near linear time algorithms for approximate matchings for the unit disk graph and the general disk graph cases. Our results are summarized in Table 1. Note, in this paper, we assume the input to our algorithms to be a set of disks.

1. Unit disk graph.
   a. Greedy matching. We show in Section 3.1 a linear time algorithm for the case of unit disks graph – this readily provides a 1/2-approximation to the maximum matching. The algorithm uses a simple grid to "capture' intersections, and then use the locality of the grid to find intersection with the remaining set of disks.
   b. $(1-\varepsilon)$-approximation. In Section 3.2 we show how to get a $(1-\varepsilon)$-approximation. The running time can be bounded by $O\big((n/\varepsilon^2)\log(1/\varepsilon)\big)$. If the diameter of union of disks is at most Δ, then the running time is $O\big(n + (\Delta^2/\varepsilon^2)\log(1/\varepsilon)\big)$.
   c. $(1-\varepsilon)$-estimation. Surprisingly, one can do even better – we show in Section 3.3 how to use importance sampling to get $(1\pm\varepsilon)$-approximation (in expectation) to the size of the maximum matching in time $O(n + \mathrm{poly}(\log n, 1/\varepsilon))$.
2. Disk graph. The general disk graph case is more challenging.
   a. Greedy matching. The greedy matching algorithm can be implemented in $O(n\log n)$ time using sweeping, see Lemma 15 (this algorithm works for any nicely behaved shapes).
   b. Approximate bipartite case. Here, we are given two sets of disks, and consider only intersections across the sets as edges. This case can be solved using range searching data-structures as was done by Efrat et al. [6] – they showed how to implement a round of the bipartite matching algorithm of Hopcroft and Karp [12] using $O(n)$ queries. It is folklore that running $O(1/\varepsilon)$ rounds of this algorithm leads to a $(1-\varepsilon)$-approximation algorithm. Coupling this with the data-structure of Kaplan et al. [13] readily leads to a near linear approximation algorithm in this case, see Section 4.3.
   c. $(1-\varepsilon)$-approximation algorithm. Surprisingly, approximate general matchings can be reduced to bipartite matchings via random coloring. Specifically, one can compute $(1-\varepsilon)$-approximate matchings using $2^{O(1/\varepsilon)}\log n$ invocations of approximate bipartite matchings algorithm mentioned above. This rather neat idea is due to Lotker et al. [14] who used in the context of parallel matching algorithms. This leads to a near linear time algorithm for $(1-\varepsilon)$-approximate matchings for disk intersection graphs, see Section 4.4 for details. The running time of the resulting algorithm is $2^{O(1/\varepsilon)} n \log^{O(1)} n$. We emphasize that this algorithm assumes nothing about the density of the input disks.
   d. $(1-\varepsilon)$-estimation. One can get an $O(n\log n)$ time estimation algorithm in this case, but one needs to assume that the input disks have "small" density ρ. Specifically, one computes a separator hierarchy and then use importance sampling on the patches, as to estimate the sampling size. The details require some care, see Section 4.6. The resulting algorithm has running time $O(n\log n)$ if the density is $o(n^{1/9})$.
3. General shapes. Somewhat surprisingly, almost all our results extends in a verbatim fashion to intersection graphs of general shapes. We need some standard assumptions about the shapes:
   a. the boundary of any pair of shapes intersects only a constant number of times,
   b. these intersections can be computed in constant time,
   c. one can compute the x-extreme and y-extreme points in a shape in constant time,
   d. one can decide in constant time if a point is inside a shape, and
   e. the boundary of a shape intersects any line a constant number of times, and they can be computed in constant time.

   For fat shapes of similar size, we also assume the diameters of all the shapes are the same up to a constant factor, and any object o contains a disk of radius $\Omega(\mathrm{diam}(o))$.

Table 1 Results. The ($\star$) indicates the result works only for disks.

Shape	Quality	Running time	Ref	Comment
Unit disks Fat shapes of similar size	$1/2$	$O(n)$	Lemma 9	Greedy
	$1-\varepsilon$	$O\left(\frac{n}{\varepsilon}\log\frac{1}{\varepsilon}\right)$	Lemma 11	
		$O\left(n+(\Delta^2/\varepsilon^2)\log\frac{1}{\varepsilon}\right)$	Remark 12	Δ = Diam. disks
	$1\pm\varepsilon$	$O(n+\varepsilon^{-6}\log^2 n)$	Theorem 14	Estimation
Unit disks	$1-\varepsilon$	$O\left(\frac{n}{\varepsilon}\log\frac{1}{\varepsilon}\right)$	Lemma 11	Approximate
Disks	$1-\varepsilon$	$O\left((n/\varepsilon)\log^{11} n\right)$	Lemma 18 ($\star$)	Bipartite
	$1-\varepsilon$	$O(2^{O(1/\varepsilon)}n\log^{12} n)$	Theorem 20 ($\star$)	General
Shapes	$1/2$	$O(n\log n)$	Lemma 15	Greedy
	$1-\varepsilon$	$O(n\log n+\frac{m}{\varepsilon}\log\frac{1}{\varepsilon})$	Lemma 16	m : # edges
	$1-\varepsilon$	$O(n\log n+\frac{n\rho}{\varepsilon}\log\frac{1}{\varepsilon})$	Lemma 17	ρ: Density
	Exact	$O(n\log n)$	Lemma 23	Matching size $O(n^{1/8})$
	$1\pm\varepsilon$	$O(n\log n+\rho^9\varepsilon^{-19}\log^2 n)$	Theorem 28	Estimation

Since the modification needed to make the algorithms work for the more general cases are straightforward, we describe the algorithms for disks.

The full version of the paper is available on the arXiv [10] and includes all missing details/proofs.

2 Preliminaries

2.1 Notations

For a graph G, let $\mathcal{M}^\star_G$ denote the maximum cardinality matching in G. Its size is denoted by $m^\star = m^\star(G) = |\mathcal{M}^\star_G|$. For a graph $G = (\mathsf{V}, \mathsf{E})$, and a set $X \subseteq \mathsf{V}$ the *induced subgraph* of G over X is $G_{|X} = \big(X, \{uv \in \mathsf{E} \mid u, v \in X\}\big)$. For a set Z, let $G - Z$ denote the graph resulting from G after deleting from it all the vertices of Z. Formally, $G - Z$ is the graph $G_{|\mathsf{V}\setminus Z}$.

▶ **Definition 1.** *For a set of objects* U*, the* ***intersection graph*** *of* U*, denoted by* $\mathcal{I}_{\mathsf{U}}$*, is the graph having* U *as its set of vertices, and there is an edge between two objects* $o, g \in \mathsf{U}$ *if they intersect. Formally,*

$$\mathcal{I}_{\mathsf{U}} = \big(\mathsf{U}, \big\{og \;\big|\; o, g \in \mathsf{U} \text{ and } o \cap g \neq \emptyset\big\}\big).$$

For a point $p \in \mathbb{R}^2$, and a set of disks $\mathcal{D}$, let

$$\mathcal{D} \sqcap p = \{\bigcirc \in \mathcal{D} \mid p \in \bigcirc\}$$

be the set of disks of $\mathcal{D}$ that contain p. Note, that the intersection graph $\mathcal{I}_p = \mathcal{I}_{\mathcal{D}\cap p}$ is a clique.

▶ **Definition 2.** *Consider a set of disks* $\mathcal{D}$*. A set* $\mathcal{D}' \subseteq \mathcal{D}$ *is an* ***independent set*** *(or simply* independent*) if no pair of disks of* $\mathcal{D}'$ *intersects.*

2.2 Low density and separators

The following is standard by now, see Har-Peled and Quanrud [9] and references therein.

▶ **Definition 3.** *A set of objects* U *in* $\mathbb{R}^d$ *(not necessarily convex or connected) has* ***density*** ρ *if any object* o *(not necessarily in* U*) intersects at most* ρ *objects in* U *with diameter equal or larger than the diameter of* o*. The minimum such quantity is denoted by* $\operatorname{density}(\mathsf{U})$*. A graph that can be realized as the intersection graph of a set of objects* U *in* $\mathbb{R}^d$ *with density* ρ *is* ρ***-dense****. The set* U *is* ***low density*** *if* $\rho = O(1)$*.*

▶ **Definition 4.** *Let* $G = (\mathsf{V}, \mathsf{E})$ *be an undirected graph. Two sets* $X, Y \subseteq \mathsf{V}$ *are* ***separate*** *in* G *if*

1. X *and* Y *are disjoint, and*
2. *there is no edge between the vertices of* X *and the vertices of* Y *in* G*.*

For a constant $\zeta \in (0,1)$*, a set* $Z \subseteq \mathsf{V}$ *is a* ζ***-separator*** *for a set* $U \subseteq \mathsf{V}$*, if* $U \setminus Z$ *can be partitioned into two* separate *sets* X *and* Y*, with* $|X| \leq \zeta |U|$ *and* $|Y| \leq \zeta |U|$*.*

▶ **Lemma 5** ([9])**.** *Let* U *be a set of* n *objects in* $\mathbb{R}^d$ *with density* ρ*. One can compute, in expected linear time, a sphere* $\mathbb{S}$ *that intersects in expectation* $\tau = O\big(\rho + \rho^{1/d} n^{1-1/d}\big)$ *objects of* U*. The sphere is computed by picking uniformly its radius from some range of the form* $[\alpha, 2\alpha]$*. Furthermore, the total number of objects of* U *strictly inside/outside* $\mathbb{S}$ *is at most* ζn*, where* ζ *is a constant that depends only on* d*. Namely, the intersection graph* $\mathcal{I}_{\mathsf{U}}$ *has a separator of size* τ *formed by all the objects of* U *intersecting* $\mathbb{S}$*.*

2.3 Importance sampling

Importance sampling is a standard technique for estimating a sum of terms. Assume that for each term in the summation, one can quickly get a coarse estimate of its value. Furthermore, assume that better estimates are possible but expensive. Importance sampling shows how to sample terms in the summation, then acquire a better estimate *only for the sampled terms*, to get a good estimate for the full summation. In particular, the number of samples is bounded independently of the original number of terms, depending instead on the coarseness of the initial estimates, the probability of success, and the quality of the final output estimate.

▶ **Lemma 6** ([2])**.** *Let* $(\mathcal{H}_1, w_1, e_1), \ldots, (\mathcal{H}_r, w_r, e_r)$ *be given, where* $\mathcal{H}_i$*'s are some structures, and* w_i *and* e_i *are numbers, for* $i = 1, \ldots, r$*. Every structure* $\mathcal{H}_i$ *has an associated weight* $\overline{\mathsf{w}}(\mathcal{H}_i) \geq 0$ *(the exact value of* $\overline{\mathsf{w}}(\mathcal{H}_i)$ *is not given to us). In addition, let* $\xi > 0$*,* γ*,* b*, and* M *be parameters, such that:*

1. $\forall i \quad w_i, e_i \geq 1$*,*
2. $\forall i \quad e_i/b \leq \overline{\mathsf{w}}(\mathcal{H}_i) \leq e_i b$*, and*
3. $\Gamma = \sum_i w_i \cdot \overline{\mathsf{w}}(\mathcal{H}_i) \leq M$*.*

Then, one can compute a new sequence of triples $(\mathcal{H}'_1, w'_1, e'_1), \ldots, (\mathcal{H}'_t, w'_t, e'_t)$*, that also complies with the above conditions, such that the estimate* $Y = \sum_{i=1}^{t} w'_i \overline{\mathsf{w}}(\mathcal{H}'_i)$ *is a multiplicative* $(1 \pm \xi)$*-approximation to* Γ*, with probability* $\geq 1 - \gamma$*. The running time of the algorithm is* $O(r)$*, and size of the output sequence is* $t = O\big(b^4 \xi^{-2} (\log\log M + \log \gamma^{-1}) \log M\big)$*.*

▶ Remark 7.

(A) The algorithm of Lemma 6 does not use the entities $\mathcal{H}_i$ directly at all. In particular, the $\mathcal{H}'_i$s are just (reweighed) copies of some original structures. The only thing that the above lemma uses is the estimates $e_1, \ldots, e_r$ and the weights $w_1, \ldots, w_r$.

(B) We are going to use Lemma 6, with $\xi = O(\varepsilon)$, $\gamma = 1/n^{O(1)}$, $b = 2$, and $M = n$. As such, the size of the output list is $L_{\text{len}} = O(\varepsilon^{-2} \log^2 n)$

2.4 Background on matchings

For a graph G, a **matching** is a set $\mathcal{C} \subseteq E(G)$ of edges, such that no pair of them not share an endpoint. A matching that has the largest cardinality possible for a graph G, is a **maximum** matching. Given a graph G and a matching $\mathcal{C}$ on G, an **alternating path** is a path with edges that alternate between matched edges (i.e., edges that are in $\mathcal{C}$) and unmatched edges (i.e., edges in $E(G) \setminus \mathcal{C}$). If both endpoints of an alternating path are unmatched (i.e., **free**), then it is an **augmenting path**. In the following, let $\mathcal{M}^\star$ denote a maximum cardinality matching in G, and let $m^\star = m^\star(G) = |\mathcal{M}^\star|$ denote its size. For $\beta \in [0,1]$, a matching $\mathcal{B} \subseteq E(G)$ is an **β-matching** (or β-approximate matching) if $|\mathcal{B}| \geq \beta m^\star$. The set of vertices covered by the matching $\mathcal{B}$ is denoted by $\mathsf{V}(\mathcal{B}) = \bigcup_{uv \in \mathcal{B}} \{u, v\}$.

The *length* of a path is the number of its edges.

The following claim is well known, see [10].

▶ **Lemma 8.** *For any $\varepsilon \in (0,1)$, if $|\mathcal{C}| < (1-\varepsilon)|\mathcal{M}^\star|$, then there are at least $(\varepsilon/2)|\mathcal{M}^\star|$ disjoint augmenting paths of $\mathcal{C}$, each of length at most $4/\varepsilon$.*

3 Approximate matchings for unit disk graph

3.1 Greedy maximal matching

In a graph G, the greedy maximal matching can be computed by repeatedly picking an edge of G, adding it to the matching,and removing the two vertices of the edges from G. We do this repeatedly until no edges remain. The resulting **greedy matching** is a maximal matching, and every maximal matching is a 1/2-approximation to the maximum matching. To avoid the maximum/maximal confusion, we refer to such a matching as a greedy matching.

▶ **Lemma 9.** *Let $\mathcal{D}$ be a set of n unit disks in the plane, where a* unit disk *has radius one. One can compute, in $O(n)$ time, a $(1/2)$-approximate matching for $\mathcal{I}_\mathcal{D}$, where $\mathcal{I}_\mathcal{D}$ is the intersection graph of the disks of $\mathcal{D}$.*

Proof. For every disk, compute all the integral grid points that it covers. Every disk covers at least one, and at most five grid points. We use hashing to compute for every grid point the disks that covers it. These lists can be computed in $O(n)$ time overall. Next, for every grid point that stores more than one disk, scan it, and break it into pairs, where every pair is reported as a matching edge, and the two disks involved are removed.

By the end of this process, we computed a partial matching $\mathcal{C}$, and we have a set $\mathcal{D}'$ of leftover disks that are not matched yet. The disks of $\mathcal{D}'$ cover every integral grid point at most once. Using the hash table one can look for intersections – for every grid point that is active (i.e., has one disk of $\mathcal{D}'$ covering it), the algorithm lookup in the hash table any disk that covers any of the 8 neighboring grid points. Each such neighboring point offers one disk that might intersect the current disk. If we find an intersecting pair, the algorithm outputs it (removing the two disks involved). This requires $O(1)$ time per active grid point, and linear time overall. At the end of this process, all the remaining disks are disjoint, implying that the computed matching is maximal and thus a $(1/2)$-approximation to the maximum matching. ◀

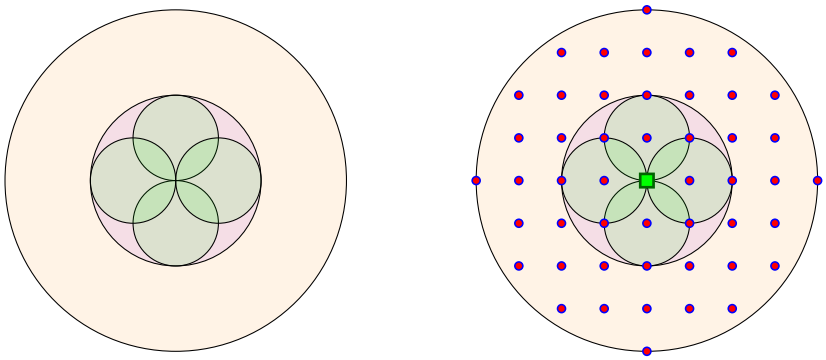


Figure 1 A tower can interact with at most 48 other towers.

3.2 $(1-\varepsilon)$-approximation

▶ **Lemma 10.** *Let $\mathcal{D}$ be a set of n unit disks, and let $\varepsilon \in (0,1)$ be a parameter. Then, one can compute, in $O\big((n/\varepsilon^2)\log(1/\varepsilon)\big)$ time, an $(1-\varepsilon)$-matching in $\mathcal{I}_{\mathcal{D}}$, where $\mathcal{I}_{\mathcal{D}}$ is the intersection graph of $\mathcal{D}$.*

If the diameter of $\cup\mathcal{D}$ is Δ, then the running time is $O\big(n + (\Delta^2/\varepsilon^3)\log(1/\varepsilon)\big)$.

Proof. Using a unit grid, the algorithm computes for each disk in $\mathcal{D}$ a grid point that it contains, and register the disk with this point. Using hashing this can be done in $O(n)$ time overall. For a grid point p, let $\ell(p)$ be the list of disks that are registered with it. Let $p_1, \ldots p_\tau$ be the points with non-empty lists.

For a point p_i, the graph $\mathcal{I}_{\mathcal{D}\cap p_i}$ is the **tower** of p_i. Consider a maximum matching $\mathcal{M}^\star$ of $\mathcal{I}_{\mathcal{D}}$. An edge $uv \in \mathcal{M}^\star$ is a **cross edge** if u and v belong to two different towers. Observe that if there are two cross edges between two towers in a matching, then we can exchange them by two edges internals to the two towers, preserving the size of the matching – this observation is due to Bonnet et al. [3]. As such, we can assume that there is at most one cross edge between any two towers in the maximum matching $\mathcal{M}^\star$. In addition, any tower can have edges only with towers in its neighborhood – specifically, two towers might have an edge between them, if the distance between their centers is at most 4. As such, the number of cross edges in $\mathcal{M}^\star$ is at most $24\tau = 48\tau/2$, as each tower interacts with at most 48 other towers, see Figure 1.

If a tower has more than (say) $200/\varepsilon$ disks in it, then we add the greedy matching in the tower to the output, and remove all the disks in the tower. This yields a matching of size at least $100/\varepsilon$, that destroys at most 48 additional edges from the optimal matching. Thus, it is sufficient to compute the approximate matching in the residual graph. We repeat this process till all the remaining towers have at most $O(1/\varepsilon)$ disks in them. Let $\mathcal{D}'$ be the remaining set of disks – by the bounded depth, each disk intersects at most $O(1/\varepsilon)$ other disks, and the intersection graph of $G = \mathcal{I}_{\mathcal{D}'}$ has $|E(G)| = O(n/\varepsilon)$ edges, and can be computed in $O(n/\varepsilon)$ time. Using the algorithm of Duan and Pettie [5] on $\mathcal{I}_{\mathcal{D}'}$, computing a $(1-\varepsilon/2)$-approximate matching takes $O(|E(G)|\varepsilon^{-1}\log\varepsilon^{-1}) = O(n\varepsilon^{-2}\log\varepsilon^{-1})$ time. It is straightforward to verify that this matching, together with the greedy matching of the "tall" towers yields that desired $(1-\varepsilon)$-approximation.

As for the running for the case that the diameter Δ is relatively small – observe that after the cleanup step, there are at most $O(\Delta^2)$ towers, and each tower contains $O(1/\varepsilon)$ disks (each disk intersects $O(1/\varepsilon)$ disks). As such, the residual graph has at most $O(\Delta^2/\varepsilon^2)$ edges, and the running time of the Duan and Pettie [5] algorithm is $O\big((\Delta^2/\varepsilon^3)\log(1/\varepsilon)\big)$. ◀

Using a reduction of Bonnet et al. [3], we can reduce the residual graph even further, resulting in a slightly faster algorithm.

▶ **Lemma 11.** *Let $\mathcal{D}$ be a set of n unit disks, and let $\varepsilon \in (0,1)$ be a parameter. One can compute, in $O\big((n/\varepsilon)\log(1/\varepsilon)\big)$ time, an $(1-\varepsilon)$-matching in $\mathcal{I}_{\mathcal{D}}$.*

▶ Remark 12. If the set of disks of $\mathcal{D}$ has diameter Δ, then the cleanup stage reduces the number of disks to $O(\Delta^2/\varepsilon)$. Then one can involve the algorithm of Lemma 11. The resulting algorithm has running time $O\big(n + (\Delta^2/\varepsilon^2)\log(1/\varepsilon)\big)$.

▶ Remark 13. The above algorithm can be modified to work in similar time for shapes of similar size – the only non-trivial step is computing the intersection graph H. This involves taking all the shapes in a tower, and its neighboring towers, computing their arrangement, and extracting the intersection pairs. If this involves ν shapes, then this takes $O(\nu^2)$ time. Each shape would be charged $O(\nu^2/\nu)$ amortized time, which results in $O(n/\varepsilon)$ time, as $\nu = O(1/\varepsilon)$. The rest of the algorithm remains the same.

3.3 Matching size estimation

▶ **Theorem 14.** *Let $\mathcal{D}$ be a set of n unit disks, and let $\varepsilon \in (0,1)$ be a parameter. One can output a number Z, such that $(1-\varepsilon)m^\star \le \mathbb{E}[Z]$ and $\mathbb{P}[Z < (1+\varepsilon)m^\star] \ge 1 - 1/n^{O(1)}$, where $m^\star$ is the size of the maximum matching in $\mathcal{I}_{\mathcal{D}}$, where $\mathcal{I}_{\mathcal{D}}$ is the intersection graph of $\mathcal{D}$. The running time of the algorithm is $O(n + \varepsilon^{-6}\log\varepsilon^{-1}\log^2 n)$.*

Proof. We randomly shift a grid of size length $\psi = \lceil 32/\varepsilon \rceil$ over the plane, by choosing a random point $p \in [0,\psi]^2$. Formally, the (i,j)th cell in this grid is $p + (i\psi, j\psi) + [0,\psi]^2$, where i, j are integers. For a pair of unit disks that intersect, with probability $\ge 1 - \varepsilon/2$ they both fall into the interior of a single grid cell. As such, throwing away all the disks that intersect the boundaries of the shifted grid, the remaining set of disks $\mathcal{D}'$, in expectation, has a matching of size at least $(1-\varepsilon/8)m^\star$.

For a grid cell $\square$ in this shifted grid, let $\mathcal{D}_\square$ be the set of disks of $\mathcal{D}$ that are fully contained in $\square$. A grid cell $\square$ is *active* if $\mathcal{D}_\square$ is not empty. Let $\mathcal{B}$ be the set of active grid cells. For a cell $\square \in \mathcal{B}$, let $m^\star_\square$ be the size of the maximum matching in $\mathcal{I}_{\mathcal{D}_\square}$. Using the algorithm of Lemma 9, compute in $O(n)$ time overall, for all $\square \in \mathcal{B}$, a number $e_\square$ such that $m^\star_\square/2 \le e_\square \le m^\star_\square$.

The task at hand is to estimate the sum $\sigma = \sum_{\square\in\mathcal{B}} m^\star_\square$, where $\sigma \le m^\star$ and $\mathbb{E}[\sigma] \ge (1-\varepsilon/8)m^\star$. To this end, we use importance sampling to reduce the number of terms in the summation of σ that need to be evaluated. Each term $m^\star_\square$ is 1/2-approximated by $e_\square$, and thus applying the algorithm of Lemma 6, to these approximation, with $\xi = \varepsilon/32$, $b = 2$, $M = n$, and $\gamma = 1/n^{10}$, we get that

$$t = O\big(b^4\xi^{-2}(\log\log M + \log\gamma^{-1})\log M\big) = O(\varepsilon^{-2}\log^2 n)$$

terms need to be evaluated (exactly if possible, but a $(1-\varepsilon/16)$-approximation is sufficient) to get $1 \pm \varepsilon/8$ estimate for σ. For each such cell, we apply the algorithm of Remark 12, to get $(1-\varepsilon/16)$-approximation. For a cell $\square$ this takes $O\big(|\mathcal{D}_\square| + (1/\varepsilon^4)\log(1/\varepsilon)\big)$ time. Summing over all these t cells, the running time is $O\big(n + (t/\varepsilon^4)\log(1/\varepsilon)\big)$. ◀

4 Approximate maximum matching for general disks

4.1 The greedy algorithm

The following 1/2-approximation algorithm works (with the same running time) for any simply connected shapes that are well-behaved.

▶ **Lemma 15.** *Let $\mathcal{D}$ be a set of n disks in the plane. One can compute a greedy matching $\mathcal{C}$ for $\mathcal{I}_{\mathcal{D}}$ in $O(n \log n)$ time. This matching $\mathcal{C}$ is a 1/2-approximation – that is, $|\mathcal{C}| \geq m^\star/2$, where $m^\star$ is the size of the maximum cardinality matching in $\mathcal{I}_{\mathcal{D}}$.*

4.2 Approximation algorithm when the graph is sparse

▶ **Lemma 16.** *Let $\mathcal{D}$ be a set of n disks in the plane such that the intersection graph $\mathcal{I}_{\mathcal{D}}$ has m edges. For a parameter $\varepsilon \in (0,1)$, one can compute, in $O\big(n \log n + (m/\varepsilon)\log(1/\varepsilon)\big)$ time, an $(1-\varepsilon)$-matching in $\mathcal{I}_{\mathcal{D}}$.*

Proof. Computing the vertical decomposition of the arrangement $\mathcal{A}(\mathcal{D})$ can be done in $O(n \log n + m)$ randomized time, using randomized incremental construction [4], as the complexity of $\mathcal{A}(\mathcal{D})$ is $O(n+m)$. This readily generates all the edges that arise out of pairs of disks with intersecting boundaries.

The remaining edges are created by one disk being enclosed completely inside another disk. One can perform a DFS on the dual graph of this arrangement, such that whenever visiting a trapezoid, the traversal maintains the set of disks that contains it. This takes time linear in the size of the arrangement, since the list of disks containing a point changes by at most one element between two adjacent faces. Now, whenever visiting a vertical trapezoid that on its non-empty vertical wall on the left contains an extreme right endpoint of a disk $\bigcirc$, the algorithm reports all the disks that contains this face, as having an edge with $\bigcirc$. Since every edge is generated at most $O(1)$ times by this algorithm, it follows that its overall running time is $O(n \log n + m)$.

Now that we computed the intersection graph, we apply the algorithm of Duan and Pettie [5]. This takes $O\big((m/\varepsilon)\log(1/\varepsilon)\big)$ time, and computes the desired matchings. ◀

The above is sufficient if the intersection graph is sparse, as is the case if the graph is low density.

▶ **Lemma 17.** *Let $\mathcal{D}$ be a set of n disks in the plane with density ρ. For a parameter $\varepsilon \in (0,1)$, one can $(1-\varepsilon)$-approximate the maximum matching in $\mathcal{I}_{\mathcal{D}}$ in $O\big(n \log n + \frac{n\rho}{\varepsilon}\log\frac{1}{\varepsilon}\big)$ time.*

Proof. The smallest disk in $\mathcal{D}$ intersects at most ρ other disks of $\mathcal{D}$. Removing this disk and repeating this argument, implies that $\mathcal{I}_{\mathcal{D}}$ has at most ρn edges. The result now readily follows from Lemma 16. ◀

4.3 The bipartite case

Consider computing maximum matching when given two sets of disks $\mathcal{D}_1, \mathcal{D}_2$, where one considers only intersections between disks that belong to different sets – that is the bipartite case. Efrat et al. [6] showed how to implement one round of Hopcroft-Karp algorithm using $O(n)$ dynamic range searching operations on a set of disks. Using the (recent) data-structure

of Kaplan et al. [13], one can implement this algorithm. Each operation on the dynamic disks data-structure takes $O(\log^{11} n)$ time. If our purpose is to get an $(1-\varepsilon)$-approximation, we need to run this algorithm $O(1/\varepsilon)$ times, so that all paths of length $O(1/\varepsilon)$ get augmented, resulting in the following.

▶ **Lemma 18.** *Given sets $\mathcal{D}_1, \mathcal{D}_2$ at most n disks in he plane, one can $(1-\varepsilon)$-approximate the maximum matching in the bipartite graph*

$$\mathcal{I}_{\mathcal{D}_1,\mathcal{D}_2} = (\mathcal{D}_1 \cup \mathcal{D}_2, \{\bigcirc_1\bigcirc_2 \mid \bigcirc_1 \in \mathcal{D}_1, \bigcirc_2 \in \mathcal{D}_2, \text{ and } \bigcirc_1 \cap \bigcirc_2 \neq \emptyset\}).$$

in $O\big((n/\varepsilon)\log^{11} n\big)$ time. Any augmenting path for this matching has length at least $4/\varepsilon$.

4.4 Approximate matching via reduction to the bipartite case

We use a reduction, due to Lotker et al. [14], of approximate general matchings to the bipartite case.

4.4.1 The Algorithm

The input is a set $\mathcal{D}$ of n disks, and a parameter $\varepsilon \in (0,1)$. The algorithm maintains a matching $\mathcal{C}$ in $\mathcal{I}_{\mathcal{D}}$. Initially, this matching can be the greedy matching. Now, the algorithm repeats the following $O(c_\varepsilon \log n)$ times, where $c_\varepsilon = 2^{8/\varepsilon}$:

***i*th iteration:** Randomly color the disks of $\mathcal{D}$ by two colors (say 1 and 2), and let $\mathcal{D}_1, \mathcal{D}_2$ be the resulting partition. Remove from $\mathcal{D}_1$ any pair of disks $\bigcirc_1, \bigcirc_2$ such that $\bigcirc_1\bigcirc_2$ is in the current matching $\mathcal{C}$. Do the same to $\mathcal{D}_2$. Let $\mathcal{C}'_i$ be edges of $\mathcal{C}$ that appear in $H_i = \mathcal{I}_{\mathcal{D}_1,\mathcal{D}_2}$. Using Lemma 18, find an $(1-\varepsilon/16)$-approximate maximum matching in H_i, and let $\mathcal{C}''_i$ be this matching. Augment $\mathcal{C}$ with the augmenting paths in $\mathcal{C}'_i \oplus \mathcal{C}''_i$.

The intuition behind this algorithm is that this process would compute all the augmenting paths of $\mathcal{C}$ of length (say) $\leq 4/\varepsilon$, which implies that the resulting matching is the desired approximation.

4.4.2 Analysis

▶ **Lemma 19.** *The above algorithm outputs a matching of size $\geq (1-\varepsilon)\boldsymbol{m}^\star$, with probability $\geq 1 - 1/n^{O(1)}$.*

4.4.3 The result

▶ **Theorem 20.** *Let $\mathcal{D}$ be a set of n disks in the plane, and $\varepsilon \in (0,1)$ be a parameter. One can compute a matching in $\mathcal{I}_{\mathcal{D}}$ of size $\geq (1-\varepsilon)\boldsymbol{m}^\star$, in $O(2^{8/\varepsilon} n \log^{12} n)$ time, where $\boldsymbol{m}^\star$ is the cardinality of the maximum matching in $\mathcal{I}_{\mathcal{D}}$. The algorithm succeeds with high probability.*

▶ Remark 21. Note, that the above algorithm does not work for fat shapes (even of similar size), since the range searching data-structure of Kaplan et al. [13] can not to be used for such shapes.

4.5 Algorithm for the case the maximum matching is small

If $n_{\mathcal{C}} = |\mathcal{C}|$ is small (say, polylogarithmic), it turns out that one can compute the *maximum* matching exactly in near linear time.

▶ **Lemma 22.** *For a set X of n disks, and any constant $\delta \in (0,1)$, one can preprocess X, in $O(n^{3+\delta} \log n)$ time, such that given a query disk $\bigcirc$, the algorithm outputs, in $O(\log n)$ time, a pointer to a (unique) list containing all the disks intersecting the query disk.*

▶ **Lemma 23.** *Let $\mathcal{D}$ be a set of n disks in the plane. Then, in $O(n \log n)$ time, one can decide if $m^\star(\mathcal{D}) = O(n^{1/8})$, and if so compute and output this maximum matching.*

4.6 Estimation of matching size using separators

The input is a set $\mathcal{D}$ of n disks in the plane with density ρ (if the value of ρ is not given, it can be approximated in near linear time [1]). Our purpose here is to $(1-\varepsilon)$-estimate the size of the maximum matching in $\mathcal{D}$ in near linear time. Since we can check (and compute it) if the maximum matching is smaller than $n^{1/8}$ by Lemma 23, in $O(n \log n)$ time, assume that the matching is bigger than that.

4.6.1 Preliminaries

▶ **Lemma 24** ([8])**.** *Let p be a point in the plane, and let r be a random number picked uniformly in an interval $[\alpha, 2\alpha]$. Let $\mathcal{H}$ be a set of interior disjoint disks in the plane. Then, the expected number of disks of $\mathcal{H}$ that intersects the circle $\bigcirc = \bigcirc(p, r)$, that is centered at p and has radius r, is $O(\sqrt{|\mathcal{H}|})$.*

4.6.2 Algorithm idea and divisions

A natural approach to our problem is to break the input set of disks into small sets, and then estimate the maximum matching size in each one of them. The problem is that for this to work, we need to partition the disks participating in the optimal matching, as this matching can be significantly smaller than the number of input disks. Since we do not have the optimal matchings, we would use a proxy to this end – the greedy matching. The algorithm recursively partitions it using a random cycle separator provided by Lemma 5. We then partition the disks into three sets – inside the cycle, intersecting the cycle (i.e., the separator), and outside the cycle. The algorithm continues this partition recursively on the in/out sets, forming a partition hierarchy.

▶ Remark 25. For a set generated by this partition, its **boundary** is the set of all disks that intersect it and are not in the set. The algorithm maintains the property that for such a set with t disks, the number of its boundary vertices is bounded by $O(\rho + \sqrt{\rho t})$. This can be ensured by alternately separating for cardinality of the set, and for the cardinality of the boundary vertices, see [9] and references therein for details. For simplicity of exposition we assume this property holds, without going into the low level details required to ensure this.

4.6.3 The algorithm

The input is a set $\mathcal{D}$ of n disks in the plane with density ρ, and parameters $\varepsilon \in (0,1)$. The algorithm computes the greedy matching, denoted by $\mathcal{C}$, using Lemma 15. If this matching is smaller than $O(n^{1/8})$, then the algorithm computes the maximum matching using Lemma 23, and returns it.

Otherwise, the algorithm partitions the disks of $\mathcal{H} = \mathsf{V}(\mathcal{C})$ recursively using separators, creating a separator hierarchy as described above. Conceptually, a subproblem here is a region R in the plane formed by the union of some faces in an arrangement of circles (i.e., the separators used in higher level of the recursion). Assume the algorithm has the sets of disks $\mathcal{H}_{\subseteq R} = \{\bigcirc \in \mathcal{H} \mid \bigcirc \subseteq R\}$ and $\mathcal{D}_{\subseteq R} = \{\bigcirc \in \mathcal{D} \mid \bigcirc \subseteq R\}$ at hand. The algorithm computes a separator of $\mathcal{H}_{\subseteq R}$, computes the relevant sets for the children, and continues recursively on the children. Thus, for a node u in this recursion tree, there is a corresponding region $R(u)$, a set of active disks $\mathcal{H}_u = \mathcal{H}_{\subseteq R(u)}$, and $\mathcal{D}_u = \mathcal{D}_{\subseteq R(u)}$.

The recursion stops the construction in node u if $|\mathcal{H}_u| \leq b$, where

$$b = \mathsf{c}_3 \rho / \varepsilon^2,$$

and c_3 is some sufficiently large constant. This implies that this recursion tree has $U = O(m^\star/b)$ leafs.

If a disk of $\mathcal{D}$ intersects some separator cycles then it is added to the set of "lost" disks $\mathcal{L}$. The hierarchy maps every disk of $\mathcal{D} \setminus \mathcal{L}$ to a leaf. As such, for every leaf u of the separator tree, there is an associated set $\mathcal{D}_u$ of disks stored there. All these leaf sets, together with $\mathcal{L}$, form a disjoint partition of $\mathcal{D}$.

The algorithm now computes for every leaf set a greedy matching, using Lemma 15. Let e_v be the size of this matching. Let Ξ be the set of all leaf nodes. The algorithm next $(1 \pm \varepsilon/4)$-estimates $\sum_{v \in \Xi} m^\star(\mathcal{D}_v)$, using importance sampling, with the estimates $e_v \leq m^\star(\mathcal{D}_v) \leq 2e_v$, for all v. Using, Lemma 6, this requires computing $(1-\varepsilon/8)$-approximate maximum matching for

$$t = O\big(2^4 \varepsilon^{-2} (\log\log n + \log n) \log n\big) = O(\varepsilon^{-2} \log^2 n)$$

leafs, this is done using the algorithm of Lemma 23 if the maximum matching is small compared to the number of disks in this subproblem, and the algorithm of Lemma 17 otherwise. The algorithm now returns the estimate returned by the algorithm of Lemma 6.

4.6.4 Analysis

▶ **Lemma 26.** *We have* $\mathbb{E}\big[\sum_{v \in L} m^\star(\mathcal{D}_v)\big] \geq (1 - \varepsilon/4) m^\star(\mathcal{D})$.

▶ **Lemma 27.** *The running time of the above algorithm is* $O(n \log n + \rho^9 \varepsilon^{-19} \log^2 n)$.

▶ **Theorem 28.** *Given a set* $\mathcal{D}$ *of* n *disks in the plane with density* ρ*, and a parameter* $\varepsilon \in (0,1)$*, one can compute in* $O(n \log n + \rho^9 \varepsilon^{-19} \log^2 n)$ *time, a number* Z*, such that* $(1-\varepsilon) m^\star \leq \mathbb{E}[Z]$ *and* $\mathbb{P}[Z > (1+\varepsilon) m^\star] < 1/n^{O(1)}$*, where* $m^\star = m^\star(\mathcal{D})$ *is the size of the maximum matching in* $\mathcal{I}_{\mathcal{D}}$.

References

1 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.

2 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *ACM Trans. Algo.*, 16(4), September 2020. doi:10.1145/3404867.

3 Édouard Bonnet, Sergio Cabello, and Wolfgang Mulzer. Maximum matchings in geometric intersection graphs. In Christophe Paul and Markus Bläser, editors, *Proc. 37th Internat. Sympos. Theoret. Asp. Comp. Sci.* (STACS), volume 154 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.31.

4 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Santa Clara, CA, USA, 3rd edition, 2008. doi:10.1007/978-3-540-77974-2.

5 Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *J. ACM*, 61(1), January 2014. doi:10.1145/2529989.

6 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.

7 Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for general graph-matching problems. *J. Assoc. Comput. Mach.*, 38(4):815–853, 1991. doi:10.1145/115234.115366.

8 Sariel Har-Peled. A simple proof of the existence of a planar separator. *ArXiv e-prints*, April 2013. arXiv:1105.0103.

9 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM J. Comput.*, 46(6):1712–1744, 2017. doi:10.1137/16M1079336.

10 Sariel Har-Peled and Everett Yang. Approximation algorithms for maximum matchings in geometric intersection graphs. *CoRR*, abs/2201.01849, 2022. arXiv:2201.01849.

11 Nicholas J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39(2):679–702, 2009. doi:10.1137/070684008.

12 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.

13 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete Comput. Geom.*, 64(3):838–904, September 2020. doi:10.1007/s00454-020-00243-7.

14 Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *J. Assoc. Comput. Mach.*, 62(5):38:1–38:17, 2015. doi:10.1145/2786753.

15 Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *Proc. 45th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 248–255. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.40.

16 Marcin Mucha and Piotr Sankowski. Maximum matchings in planar graphs via Gaussian elimination. *Algorithmica*, 45(1):3–20, 2006. doi:10.1007/s00453-005-1187-5.

17 Raphael Yuster and Uri Zwick. Maximum matching in graphs with an excluded minor. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proc. 18th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 108–117. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283396.