Maximum Matchings in Geometric Intersection Graphs

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France
edouard.bonnet@ens-lyon.fr

Sergio Cabello
Faculty of Mathematics and Physics, University of Ljubljana, and IMFM, Slovenia
sergio.cabello@fmf.uni-lj.si

Wolfgang Mulzer
Institut für Informatik, Freie Universität Berlin, Germany.
mulzer@inf.fu-berlin.de

Abstract

Let G be an intersection graph of n geometric objects in the plane. We show that a maximum matching in G can be found in $O(\rho^{3\omega/2}n^{\omega/2})$ time with high probability, where ρ is the density of the geometric objects and $\omega > 2$ is a constant such that $n \times n$ matrices can be multiplied in $O(n^{\omega})$ time.

The same result holds for any subgraph of G, as long as a geometric representation is at hand. For this, we combine algebraic methods, namely computing the rank of a matrix via Gaussian elimination, with the fact that geometric intersection graphs have small separators.

We also show that in many interesting cases, the maximum matching problem in a general geometric intersection graph can be reduced to the case of bounded density. In particular, a maximum matching in the intersection graph of any family of translates of a convex object in the plane can be found in $O(n^{\omega/2})$ time with high probability, and a maximum matching in the intersection graph of a family of planar disks with radii in $[1, \Psi]$ can be found in $O(\Psi^6 \log^{11} n + \Psi^{12\omega} n^{\omega/2})$ time with high probability.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Graph algorithms analysis

Keywords and phrases computational geometry, geometric intersection graph, maximum matching, disk graph, unit-disk graph

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.27

Related Version A full version of the paper is available at `http://arxiv.org/abs/1910.02123`.

Funding *Sergio Cabello*: Supported by the Slovenian Research Agency (P1-0297, J1-9109, J1-8130, J1-8155).
Wolfgang Mulzer: Supported in part by ERC StG 757609.

1 Introduction

Let $\mathcal{U}$ be a family of (connected and compact) objects in $\mathbb{R}^2$. The ***intersection graph*** $G_{\mathcal{U}}$ of $\mathcal{U}$ is the undirected graph with vertex set $\mathcal{U}$ and edge set

$$E(G_{\mathcal{U}}) \;=\; \{UV \mid U, V \in \mathcal{U},\, U \cap V \neq \emptyset\}.$$

If the objects in $\mathcal{U}$ are partitioned into two sets, one can also define the *bipartite* intersection graph, a subgraph of $G_{\mathcal{U}}$, in the obvious way. Consider the particular case when $\mathcal{U}$ is a set of disks. Then, we call $G_{\mathcal{U}}$ a ***disk graph***, and if all disks in $\mathcal{U}$ have the same radius, a ***unit-disk graph***. Unit disk graphs are often used to model ad-hoc wireless communication networks and sensor networks [11, 14, 29]. Disks of varying sizes and other shapes become relevant when different sensors cover different areas. Moreover, general disk graphs serve as a tool to approach other problems, like the barrier resilience problem [16].

We consider a classic optimization problem, *maximum matching*, in the setting of geometric intersection graphs, and introduce two new techniques, each interesting in its own. First, we provide an efficient algorithm to compute a maximum matching in any subgraph of the intersection graph of geometric objects of low density. Second, we provide a sparsification technique to reduce the maximum matching problem in a geometric intersection graph to the case of low density. The sparsification works for convex shapes of similar sizes for which certain range searching operations can be done efficiently.

In this paper, we use ω to denote a constant such that $\omega > 2$ and any two $n \times n$ matrices can be multiplied in time $O(n^\omega)$.[1]

Maximum matching in intersection graphs of geometric objects of low density.

We first introduce some geometric concepts. The diameter of a set $X \subset \mathbb{R}^2$, denoted by $\operatorname{diam}(X)$, is the supremum of the distances between any two points of X. The ***density*** $\rho(\mathcal{U})$ of a family $\mathcal{U}$ of objects is

$$\rho(\mathcal{U}) \;=\; \max_{X \subseteq \mathbb{R}^2} \big|\{U \in \mathcal{U} \mid \operatorname{diam}(U) \geq \operatorname{diam}(X),\, U \cap X \neq \emptyset\}\big|. \tag{1}$$

One can also define the density by considering for X only disks. Since an object of diameter d can be covered by $O(1)$ disks of diameter d, this changes the resulting parameter by only a constant. (See, for example, the book by de Berg et al. [6, Section 12.5] for such a definition.) The ***depth*** (ply) of $\mathcal{U}$ is the largest number of objects that cover a single point:

$$\max_{p \in \mathbb{R}^2} \big|\{U \in \mathcal{U} \mid p \in U\}\big|.$$

For disk graphs and square graphs, the depth and the density are linearly related; see for example Har-Peled and Quanrud [13, Lemma 2.7]. More generally, bounded depth and bounded density are equivalent whenever we consider homothets of a constant number of shapes. Density and depth are usually considered in the context of realistic input models; see de Berg et al. [7] for a general discussion.

Let $\mathbb{G}_\rho$ be the family of *subgraphs* of intersection graphs of geometric objects in the plane with density at most ρ. Our goal is to compute a maximum matching in graphs of $\mathbb{G}_\rho$,

[1] In the literature, it is more common to assume $\omega \geq 2$. We adopt the stronger assumption $\omega > 2$ because it simplifies the bounds. If $\omega = 2$ is allowed, then the running times that we state have additional logarithmic factors.

assuming the availability of a geometric representation of the graph and a few basic geometric primitives on the geometric objects. For this, we consider the density ρ as an additional parameter. Naturally, the case $\rho = O(1)$ of *bounded density* is of particular interest.

In a general graph $G = (V, E)$ with n vertices and m edges, the best running time for computing a maximum matching in G depends on the ratio m/n. The classic algorithm of Micali and Vazirani [20, 27] is based on augmenting paths, and it finds a maximum matching in $O(\sqrt{n}m)$ time. Mucha and Sankowski [22] use algebraic tools to achieve running time $O(n^\omega)$. More recently, Mądry [19] showed that an approach through interior-point methods yields an algorithm with running time roughly $O(m^{10/7})$. As we shall see, for $G \in \mathbb{G}_\rho$, we have $m = O(\rho n)$, and this bound is asymptotically tight. Thus, for $G \in \mathbb{G}_\rho$, the running times of these three algorithms become $O(\rho n^{3/2})$, $O(n^\omega)$ and $O\big((\rho n)^{10/7}\big)$, respectively.

There is a specialized algorithm for certain classes of *bipartite* geometric intersection graphs. Efrat, Itai, and Katz [9] show how to compute the maximum matching in bipartite unit disk graphs in $O(n^{3/2} \log n)$ time. Having bounded density does not help in this algorithm; it has $O(\sqrt{n})$ rounds, each of which needs $\Omega(n)$ time. The same approach can be used for other geometric shapes if a certain semi-dynamic data structure is available. In particular, using the data structure of Kaplan et al. [15] for additively-weighted nearest neighbors, finding a maximum matching in a bipartite intersection graph of disks takes $O(n^{3/2} \operatorname{polylog} n)$ time. We are not aware of any similar results for non-bipartite geometric intersection graphs.

We show that a maximum matching in a graph of $\mathbb{G}_\rho$ with n vertices can be computed in $O(\rho^{3\omega/2} n^{\omega/2}) = O(\rho^{3.56} n^{1.19})$ time. The algorithm is randomized and succeeds with high probability. It uses the algebraic approach by Mucha and Sankowski [23] for planar graphs with an extension by Yuster and Zwick [28] for H-minor-free graphs. As noted by Alon and Yuster [4], this approach works for *hereditary*[2] graph families with bounded average degree and small separators. We note that the algorithm can be used for graphs of $\mathbb{G}_\rho$, because we have average degree $O(\rho)$ and balanced separators of size $O(\sqrt{\rho n})$ [13, 25]. However, finding the actual dependency on ρ is difficult because it plays a role in the average degree, in the size of the separators, and the algorithm has a complex structure with several subroutines that must be distilled.

There are several noteworthy features in our approach. For one, we solve a geometric problem using linear algebra, namely Gaussian elimination. The use of geometry is limited to finding separators, bounding the degree, and constructing the graph explicitly. Note that the role of subgraphs in the definition of $\mathbb{G}_\rho$ is a key feature in our algorithm. On the one hand, we need a hereditary family of graphs, as needed to apply the algorithm. On the other hand, it brings more generality; for example, it includes the case of bipartite graphs defined by colored geometric objects.

Compared to the work of Efrat, Itai, and Katz [9], our algorithm is for arbitrary subgraphs of geometric intersection graphs, not only bipartite ones; it works for any objects, as it does not use advanced data structures that may depend on the shapes. On the other hand, it needs the assumption of low density. Compared to previous algorithms for arbitrary graphs and ignoring polylogarithmic factors, our algorithm is faster when $\rho = o(n^{(20-7\omega)/(21\omega-20)})$. Using the current bound $\omega < 2.373$, this means that our new algorithm is faster for $\rho = O(n^{0.113})$.

Our matching algorithm also applies for intersection graphs of objects in 3-dimensional space. However, in this case there is no algorithmic gain with the current bounds on ω: one gets a running time of $O(n^{2\omega/3})$ when $\rho = O(1)$, which is worse than constructing the graph

[2] closed under taking subgraphs

explicitly and using the algorithm of Micali and Vazirani or the algorithm of Mądry.

Sparsification – Reducing to bounded depth.

Consider a family of convex geometric objects $\mathcal{U}$ in the plane where each object contains a square of side length 1 and is contained in a square of side length $\Psi \geq 1$. Our objective is to compute a maximum matching in the intersection graph $G_{\mathcal{U}}$.[3] Our goal is to transform this problem to finding a maximum matching in the intersection graph of a subfamily $\mathcal{U}' \subset \mathcal{U}$ with bounded depth. Then we can employ our result from above for $G_{\mathcal{U}'}$ or, more generally, any algorithm for maximum matching (taking advantage of the sparsity of the new instance).

We describe a method that is fairly general and works under comparatively mild assumptions and also in higher dimensions. However, for an efficient implementation, we require that the objects under considerations support certain range searching operations efficiently. We discuss how this can be done for disks of arbitrary sizes, translates of a fixed convex shape in the plane, axis-parallel objects in constant dimension, and (unit) balls in constant dimension. In all these cases, we obtain a subquadratic time algorithm for finding a maximum matching, assuming that Ψ is small. We mostly focus on the planar case, mentioning higher dimensions as appropriate.

As particular results to highlight, we show that a maximum matching in the intersection graph of any family of translates of a convex object in the plane can be found in $O(n^{\omega/2})$ time with high probability, and a maximum matching in the intersection graph of a family of planar disks with radii in $[1, \Psi]$ can be found in $O(\Psi^6 \log^{11} n + \Psi^{12\omega} n^{\omega/2})$ time with high probability.

Organization.

We begin with some general definitions and basic properties of geometric intersection graphs (Section 2). Then, in the first part of the paper, we present the new algorithm for finding a maximum matching in geometric intersection graphs of low density (Section 3). In the second part, we present our sparsification method. This is done in two steps. First, we describe a generic algorithm that works for general families of shapes that have roughly the same size, assuming that certain geometric operations can be performed quickly. (Section 4). Second, we explain how to implement these operations for several specific shape families, e.g., translates of a given convex objects and disks of bounded radius ratio (Section 5). The two parts are basically independent, where the second part uses the result from the first part as a black box, to state the desired running times. All the proofs are deferred to the long version [5].

2 Basics of (geometric intersection) graphs

Geometric objects.

Several of our algorithms work under fairly weak assumptions on the geometric input: we assume that the objects in $\mathcal{U}$ have *constant description complexity*. This means that the boundary of each object is a continuous closed curve whose graph is a semialgebraic set, defined by a constant number of polynomial equalities and inequalities of constant maximum

[3] Note that here we do not consider subgraphs of $G_{\mathcal{U}}$; we need the whole subgraph $G_{\mathcal{U}}$.

degree. For later algorithms we restrict attention to some particular geometric objects, like disks or squares.

To operate on $\mathcal{U}$, we require that our computational model supports primitive operations that involve a constant number of objects of $\mathcal{U}$ in constant time, e.g., finding the intersection points of two boundary curves; finding the intersection points between a boundary curve and a disk or a vertical line; testing whether a point lies inside, outside, or on the boundary of an object; decomposing a boundary curve into x-monotone pieces, etc. See, e.g., [15] for a further discussion and justification of these assumptions.

We emphasize that in addition to the primitives on the input objects, we do not require any special constant-time operations. In particular, even though our algorithms use algebraic techniques such as fast matrix multiplication or Gaussian elimination, we rely only on algebraic operations over $\mathbb{Z}_p$, where $p = \Theta(n^4)$. Thus, when analyzing the running time of our algorithms, we do not need to worry about the bit complexity of these operations.

Geometric intersection graphs.

The following well-known lemma bounds $|G_{\mathcal{U}}|$ in terms of ρ, and the time to construct $G_{\mathcal{U}}$.

▶ **Lemma 1.** *If $\mathcal{U}$ has n objects and density ρ, then $G_{\mathcal{U}}$ has at most $(\rho - 1)n$ edges (this holds in any dimension). If $\mathcal{U}$ consists of objects in the plane, then $G_{\mathcal{U}}$ can be constructed in $O(\rho n \log n)$ time.*

Separators in geometric intersection graphs.

The classic *planar separator theorem* by Lipton and Tarjan [8, 18] shows that any planar graph can be decomposed in a balanced way by removing a small number of vertices. Even though geometric intersection graphs can be far from planar, similar results are also available for them. These results are usually parameterized by the *depth* of the arrangement or by the *area* of the separator and the components [3, 10, 21]. The following recent result provides a small separator for general intersection graphs of bounded density.

▶ **Theorem 2** (Lemma 2.21 in [13])**.** *Let $\mathcal{U}$ be a set of n objects in $\mathbb{R}^2$ with density ρ. In $O(n)$ expected time, we can find a circle $\mathbb{S}$ such that $\mathbb{S}$ intersects at most $c\sqrt{\rho n}$ objects of $\mathcal{U}$, the exterior of $\mathbb{S}$ contains at most αn elements of $\mathcal{U}$, and the interior of $\mathbb{S}$ contains at most αn elements of $\mathcal{U}$. Here $0 < c$ and $0 < \alpha < 1$ are universal constants, independent of ρ and n.*

The proof of Theorem 2 goes roughly as follows: Pick a point in each object of $\mathcal{U}$, compute the smallest circle $\mathbb{S}'$ (or an approximation thereof) that contains, say, $n/20$ points, and then take a concentric scaled copy $\mathbb{S}$ of $\mathbb{S}'$, with scale factor uniformly at random in $[1, 2]$. With constant probability, the circle $\mathbb{S}'$ has the desired property. This can be checked easily in linear time by determining which objects of $\mathcal{U}$ are inside, outside, or intersected by $\mathbb{S}$. In expectation, a constant number of repetitions is needed to obtain the desired circle.

A family $\mathbb{G}$ of graphs is *hereditary* if for every $G \in \mathbb{G}$, it holds that all subgraphs H of G are also in $\mathbb{G}$. By definition, our family $\mathbb{G}_\rho$ of subgraphs of geometric intersection graphs with density ρ is hereditary. A graph G is ***δ-sparse*** if every subgraph H of G has at most $\delta|V(H)|$ edges. Lemma 1 implies that all graphs in $\mathbb{G}_\rho$ are ρ-sparse.

Consider a graph G and a vertex v of G. A ***vertex split*** at v consists of adding a pendant 2-path $vv'v''$, where v' and v'' are new vertices, and possibly replacing some edges uv incident to v by new edges uv''; see Figure 1 for a sequence of splits. We note that a vertex split may not replace any edges. In this case, we are just adding a pendant path of length 2.

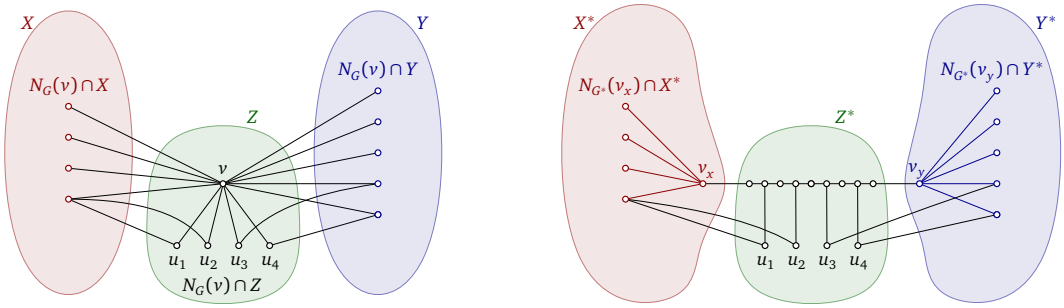

Figure 1 Splitting one single vertex of Z.

Let G' be a graph obtained from G by a sequence of k vertex splits. Then, the size of a maximum matching in G' is the size of a maximum matching in G plus k. Furthermore, from a maximum matching in G', we can easily obtain a maximum matching in G in $O(|V(G)| + |E(G)| + k)$ time. We will use vertex splits to ensure that the resulting graphs have bounded degree and a vertex set of a certain cardinality. (A vertex split may change the density, but that will not be important.) Note that if we perform a vertex split at v in a graph of $\mathbb{G}_\rho$, in general we obtain a graph of $\mathbb{G}_{\rho+2}$ because we can represent it by making two new copies of the object corresponding to v. Nevertheless, this increase in the density will not be problematic in our algorithm.

3 Maximum matching in low-density geometric intersection graphs

3.1 Separators and separator trees

A graph G has a (k, α)-separation if $V(G)$ can be partitioned into three pairwise disjoint sets X, Y, Z such that $|X \cup Z| \leq \alpha|V|$, $|Y \cup Z| \leq \alpha|V|$, $|Z| \leq k$, and such that there is no edge with one endpoint in X and one endpoint in Y. We say that Z *separates* X and Y. At the cost of making the constant α larger, we can restrict our attention to graphs of a certain minimum size.

Theorem 2 gives a $(c\sqrt{\rho n}, \alpha')$-separation for every graph of $\mathbb{G}_\rho$, for some constant $\alpha' < 1$. (A separator in $G_\mathcal{U}$ is a separator in each subgraph of $G_\mathcal{U}$.) Furthermore, such a separation can be computed in expected linear time, if the objects defining the graph are available.

A recursive application of separations can be represented as a binary rooted tree. We will use so-called (weak) *separator trees*, where the separator does not go into the subproblems. In such a tree, we store the separator at the root and recurse on each side to obtain the subtrees. We want to have small separators and balanced partitions at each level of the recursion, and we finish the recursion when we get to problems of a certain size. This leads to the following definition. Let $\gamma > 0$, $0 < \beta < 1$, and $0 < \alpha < 1$ be constants. We say that a graph G has a (γ, β, α)-***separator tree*** if there is a rooted binary full tree T with the following properties:

- (i) Each node $t \in T$ is associated with some set $Z_t \subseteq V(G)$.
- (ii) The sets Z_t, $t \in T$, partition $V(G)$, i.e., $\bigcup_{t \in T} Z_t = V(G)$, and $Z_t \cap Z_{t'} = \emptyset$, for distinct $t, t' \in T$.
- (iii) For each node $t \in T$, let $V_t = \bigcup_s Z_s$, where s ranges over the descendants of t (including t). Note that if t is an internal node with children u and v, then V_t is the disjoint union of Z_t, V_u, and V_v. If t is a leaf, then $V_t = Z_t$.

- (iv) For each internal node $t \in T$ with children u and v, (V_u, V_v, Z_t) is a $(\gamma m^\beta, \alpha)$-separation for $G[V_t]$, the subgraph of G induced by V_t, where $m = |V_t| = |Z_t| + |V_u| + |V_v|$.
- (v) For each leaf $t \in T$, we have $|V_t| \leq \Theta(\gamma^{1/(1-\beta)})$. We have chosen the size so that V_t is a $(\gamma |V_t|^\beta, \alpha)$-separator for the whole induced subgraph $G[V_t]$.

Yuster and Zwick [28] provide an algorithm that computes a separator tree of some split graph for a given graph from an H-minor-free family. As Alon and Yuster [4, Lemma 2.13] point out, this algorithm actually works for any δ-sparse hereditary graph family, as long as δ is constant. Thus, the result applies to $\mathbb{G}_\rho$. We revise the construction to make the dependency on ρ explicit.

▶ **Lemma 3.** *Given a graph G of $\mathbb{G}_\rho$ with n vertices, we can compute in $O(\rho n \log n)$ expected time a vertex-split graph G' of G and a separator tree T' for G' with the following properties:*

- *(i) the graph G' has $\Theta(\rho n)$ vertices and edges;*
- *(ii) the maximum degree of G' is at most 4;*
- *(iii) T' is a $(\gamma = O(\rho), \beta = 1/2, \alpha)$-separator tree for G', where $\alpha < 1$ is a constant (independent of ρ and n).*

Note that the split graph G' in Lemma 3 is not necessarily in $\mathbb{G}_\rho$. It is a subgraph of an intersection graph, but since we introduce copies of geometric objects when we split vertices, the density increases. In any case, this does not matter because G' will be accessed through the separator tree T'.

3.2 Nested dissection

We will need to compute with matrices. The arithmetic operations take place in $\mathbb{Z}_p$, where $p = \Theta(n^4)$ is prime. Thus, we work with numbers of $O(\log n)$-bits, and we simply need to bound the number of arithmetic operations. Using a word-RAM model of computation, each arithmetic operation needs constant time.

Let A be an $n \times n$ matrix. A Gaussian elimination step on row i is the following operation: for $j = i + 1, \ldots, n$, add an appropriate multiple of row i to row j so that the element at position (j, i) becomes 0. Elimination on row i can be performed if the entry at position (i, i) is nonzero. Gaussian elimination on A consists of performing Gaussian elimination steps on rows $i = 1, \ldots, n - 1$. This is equivalent to computing an LU decomposition of A, where L is a lower triangular matrix with units along the diagonal, and U is an upper triangular matrix. Gaussian elimination is performed *without pivoting* if, for all i, when we are about to do a Gaussian elimination step on row i, the entry at position (i, i) is non-zero. If Gaussian elimination is performed without pivoting, then the matrix is non-singular. (Pivoting is permuting the rows to ensure that the entry at position (i, i) is non-zero.)

Let $[n] = \{1, \ldots, n\}$. The ***representing graph*** $G(A)$ of an $n \times n$ matrix $A = (a_{i,j})_{i,j \in [n]}$ is

$$G(A) = \left([n], \left\{ ij \in \binom{[n]}{2} \mid a_{i,j} \neq 0 \text{ or } a_{j,i} \neq 0 \right\} \right).$$

Let T be a separator tree for $G(A)$. The row order of A is ***consistent*** with T if, whenever t' is an ancestor of t, all the rows of Z_t are before any row of $Z_{t'}$. We may assume that all the rows of Z_t are consecutive. In particular, if the rows are ordered according to a post-order traversal of T, then the row order of A is consistent with T. A careful but simple revision of the nested dissection algorithm by Gilbert and Tarjan [12] leads to the following theorem.

▶ **Theorem 4.** *Let A be an $n \times n$ matrix such that the representing graph $G(A)$ has bounded degree and assume that we are given a (γ, β, α)-separator tree T for $G(A)$, were $\gamma > 0$, $0 < \alpha < 1$, and $1/2 < \beta < 1$ are constants. Furthermore, assume that the row order of A is consistent with T and that Gaussian elimination on A is done without pivoting. We can perform Gaussian elimination (without pivoting) on A and find a factorization $A = LU$ of A in $O(\gamma^\omega n^{\beta\omega})$ time, where L is a lower triangular matrix with units along the diagonal and U is an upper triangular matrix.*

To prove Theorem 4, we need the following folklore lemma.

▶ **Lemma 5.** *Let A be an $n \times n$ matrix, and $k \leq n$. Suppose that Gaussian elimination on the first k rows of A needs no pivoting. Then, we can perform Gaussian elimination on the first k rows of A with $O(n^2 k^{\omega-2})$ arithmetic operations.*

Remark 1: Mucha and Sankowski [23] noted that the result holds when $G(A)$ is planar or, more generally, has recursive separators, using the approach by Lipton, Rose, and Tarjan [17] for nested dissection. This approach is based on the *strong* separator tree. Alon, Yuster, and Zwick [4, 28] showed that a similar result holds for graphs of bounded degree with recursive separators if one instead uses the nested dissection given by Gilbert and Tarjan [12]. In this case, we need bounded degree, but a weak separator tree suffices. Again, since we want to make the dependency on ρ explicit and since the analysis in terms of matrix multiplication time does not seem to be written down in detail anywhere, we revise the method carefully.

Remark 2: Usually, the result is stated for symmetric positive definite matrices. Reindexing a symmetric positive definite matrix gives another symmetric positive definite matrix, and Gaussian elimination on such matrices can always be performed without pivoting. Thus, for positive semidefinite matrices, we do not need to assume that the row order is consistent with T because we can reorder the rows to make it consistent with T. However, Mucha and Sankowski [23] do need the general statement in their Section 4.2, and they mention this general case after their Theorem 13. Actually, they need it over $\mathbb{Z}_p$, where the concept of positive definiteness is not even defined!

3.3 The algorithm

Assume we have a graph G of $\mathbb{G}_\rho$ with n vertices and a geometric representation, i.e., geometric objects $\mathcal{U}$ of density at most ρ such that G is a subgraph of $G_\mathcal{U}$. We want to compute a maximum matching for G. For this, we adapt the algorithm of Mucha and Sankowski [23]. We provide an overview of the approach, explain the necessary modifications, and emphasize the dependency on ρ in the different parts of the algorithm.

Using Lemma 3, we get in $O(\rho n \log n)$ expected time a vertex-split graph G' of G and a separator tree T' for G' such that:

- (i) the graph G' has $\Theta(\rho n)$ vertices and edges;
- (ii) the maximum degree of G' is at most 4;
- (iii) T' is a $(\gamma = O(\rho), \beta = 1/2, \alpha)$-separator tree for G', where $\alpha < 1$ is a constant (independent of ρ and n).

Since G' is obtained from G by vertex splits, it suffices to find a maximum matching in G'. We set $m = |V(G')| = \Theta(\rho n)$, and we label the vertices of G' from 1 to m. We consider the variables $X = (x_{ij})_{ij \in E(G')}$; i.e., each edge ij of G defines a variable x_{ij}. Consider the

$m \times m$ symbolic matrix $A[X] = A[X](G')$, defined as follows:

$$(A[X])_{i,j} = \begin{cases} x_{ij}, & \text{if } ij \in E(G') \text{ and } i < j, \\ -x_{ij}, & \text{if } ij \in E(G') \text{ and } j < i, \\ 0 & \text{otherwise.} \end{cases}$$

The symbolic matrix $A[X]$ is usually called the *Tutte matrix* of G'. It is known [24] that the rank of $A[X]$ is twice the size of the maximum matching in G'. In particular, G' has a perfect matching if and only if $\det(A[X])$ is not identically zero. Take a prime $p = \Theta(n^4)$, and substitute each variable in $A[X]$ with a value from $\mathbb{Z}_p$, each chosen independently uniformly at random. Let A be the resulting matrix. Then, with high probability, $\operatorname{rank}(A) = \operatorname{rank}(A[X])$, where on both sides we consider the rank over the field $\mathbb{Z}_p$.

From maximum matching to perfect matching.

Let $B = AA^T$. Then, B is symmetric, and the rank of B equals the rank of A. Note that $(B)_{i,j}$ is nonzero only if i and j share a neighbor in G'. Since G' has bounded degree, from the separator tree T' for G', we can obtain a separator tree T_B for the representing graph $G(B)$. Since T' was a $(\gamma = O(\rho), \beta = 1/2, \alpha)$-separator tree for G', T_B is a $(\gamma = O(\rho), \beta = 1/2, \alpha)$-separator tree for $G(B)$, where the constant hidden in $O(\rho)$ is increased by the maximum degree in G'. Using Theorem 4, we obtain that Gaussian elimination can be done in B in $O(\gamma^\omega m^{\omega/2}) = O(\rho^\omega (\rho n)^{\omega/2}) = O(\rho^{3\omega/2} n^{\omega/2})$ time, assuming that pivoting is not needed.

Mucha and Sankowski [23, Section 5] show how Gaussian elimination without pivoting can be used in B to find a collection of indices $W \subseteq [m]$ such that the centered matrix $(B)_{W,W}$, defined by rows and columns of B with indices in W, has the same rank as B. It follows that $\operatorname{rank}(A_{W,W}) = \operatorname{rank}(B_{W,W})$ and therefore $G'[W]$ contains a maximum matching of G' that is a perfect matching in $G'[W]$ (with high probability). The key insight to find such W is that, if during Gaussian elimination in B we run into a 0 along the diagonal, then the whole row and column are 0, which means that they can be removed from the matrix without affecting the rank. We summarize.

▶ **Lemma 6.** *In time $O(\rho^{3\omega/2} n^{\omega/2})$ we can find a subset W of vertices of G' such that, with high probability, $G'[W]$ has a perfect matching that is a maximum matching in G'.*

From now on, we can assume that G' has a perfect matching. We keep denoting by T' its separator tree, by A the matrix after substituting values of $\mathbb{Z}_p$ into $A[X]$, and by B the matrix AA^T. (We can compute the tree T' anew or we can reuse the same separator tree restricted to the subset of vertices.) Let Z_r denote the set stored at the root r of T'. Thus, Z_r is the first separator on G'. Let N_r be the set Z_r together with its neighbors in G'. Because G' has bounded degree, we have $|N_r| = O(|Z_r|) = O(\rho m^{1/2}) = O(\rho^{3/2} n^{1/2})$.

Mucha and Sankowski show how to compute with $O(1)$ Gaussian eliminations a matching M' in G' that covers all the vertices of Z_r and is contained in some perfect matching of G'. There are two ingredients for this. The first ingredient is to use Gaussian elimination on the matrix $B = AA^T$ to obtain a decomposition $AA^T = LDL^T$, and then use (partial) Gaussian elimination on a matrix C composed of $L_{[m],N_r}$ and $A_{N_r,[m]\setminus N_r}$ to compute $(A^{-1})_{N_r,N_r}$. (Note that in general $(A^{-1})_{N_r,N_r}$ is different from $(A_{N_r,N_r})^{-1}$. Computing the latter is simpler, while computing the former is a major insight by Mucha and Sankowski [23, Section 4.2].) Interestingly, T' is also a separator tree for the representing graph of this matrix C, and Gaussian elimination can be performed without pivoting. Thus, we can obtain in $O(\rho^\omega m^{\omega/2}) = O(\rho^{3\omega/2} n^{\omega/2})$ time the matrix $(A^{-1})_{N_r,N_r}$. The second ingredient is that, once

we have $(A^{-1})_{N_r,N_r}$, we can compute for any matching M' contained in $G'[N_r]$ a maximal (with respect to inclusion) submatching M' that is contained in a perfect matching of G'. This is based on an observation by Rabin and Vazirani [24] that shows how to find edges that belong to some perfect matching using the inverse matrix, and Gaussian elimination on the matrix $(A^{-1})_{N,N}$ to identify subsets of edges that together belong to some perfect matching. The matrix $(A^{-1})_{N_r,N_r}$ is not necessarily represented by a graph with nice separators, but it is of size $|N_r| \times |N_r|$. Thus, Gaussian elimination in $(A^{-1})_{N_r,N_r}$ takes $O(|N_r|^\omega) = O(\rho^{3\omega/2} n^{\omega/2})$ time [23, Section 2.4].

Since the graph G' has bounded maximum degree, making $O(1)$ iterations of finding a maximal matching M' in $G'[N_r]$, followed by finding a maximal subset M'' of M' contained in a perfect matching of G', and removing the vertices contained in M' plus the edges of $M' \setminus M''$, gives a matching M_* that covers Z_r and is contained in a perfect matching of G'; see [23, Section 4.3]. The vertices of M_* can be removed, and we recurse on both sides of $G' - V(M_*) \subset G' - Z_r$ using the corresponding subtrees of T'. The running time is $T(n) = O(\rho^{3\omega/2} n^{\omega/2}) + T(n_1) + T(n_2)$, where $n_1, n_2 \leq \alpha n$. This solves to $T(n) = O(\rho^{3\omega/2} n^{\omega/2})$ because $\omega/2 > 1$. We summarize in the following result. If only the family $\mathcal{U}$ is given, first we use Lemma 1 to construct $G_{\mathcal{U}}$.

▶ **Theorem 7.** *Given a graph G of $\mathbb{G}_\rho$ with n vertices together with a family $\mathcal{U}$ of geometric objects with density ρ such that G is a subgraph of $G_{\mathcal{U}}$, we can find in $O(\rho^{3\omega/2} n^{\omega/2})$ time a matching in G that, with high probability, is maximum. In particular, for a family $\mathcal{U}$ of n geometric objects with density ρ, a maximum matching in $G_{\mathcal{U}}$ can be found in $O(\rho^{3\omega/2} n^{\omega/2})$ time. The same holds for the bipartite or k-partite version of $G_{\mathcal{U}}$.*

4 Sparsification

Let $\mathcal{U}$ be a family of convex geometric objects in the plane such that each object of $\mathcal{U}$ contains a square of side length 1 and is contained in a square of side length $\Psi \geq 1$. Through the discussion we will treat Ψ as a parameter. Our objective is to reduce the problem of computing a maximum matching in the intersection graph $G_{\mathcal{U}}$ to the problem of computing a maximum matching in $G_{\mathcal{W}}$ for some $\mathcal{W} \subseteq \mathcal{U}$ of small depth.

Let $P = \mathbb{Z}^2$ be the points in the plane with integer coordinates. Each square of unit side length contains at least one point of P and each square of side length Ψ contains at most $(1+\Psi)^2 = O(\Psi^2)$ points of P. In particular, each object $U \in \mathcal{U}$ contains at least one and at most $O(\Psi^2)$ points from P.

First we provide an overview of the idea. The objects intersected by a point $p \in P$ define a clique, and thus any even number of them defines a perfect matching. We show that, for each $p \in P$, it suffices to keep a few objects pierced by p, and we show how to obtain such a suitable subfamily. The actual number of objects to keep depends on Ψ, and whether the actual computation can be done efficiently depends on the geometric shape of the objects.

For each object $U \in \mathcal{U}$, we find the lexicographically smallest point in $P \cap U$. We assume that we have a primitive operation to compute $P \cap U$ for each object $U \in \mathcal{U}$ in $O(1 + |P \cap U|) = O(\Psi^2)$ time. A simple manipulation of these incidences allows us to obtain the ***clusters***

$$\mathcal{U}_p \;=\; \{U \in \mathcal{U} \mid p \text{ lexicographically minimum in } P \cap U\}, \qquad \text{for all } p \in P.$$

Note that the clusters $\mathcal{U}_p$, for $p \in P$, form a partition of $\mathcal{U}$. This will be useful later. Clearly, the subgraph of $G_{\mathcal{U}}$ induced by $\mathcal{U}_p$ is a clique, for each $p \in P$.

We will use the usual notation

$$E(\mathcal{U}_p,\mathcal{U}_q) \;=\; \{UV \mid U \in \mathcal{U}_p,\, V \in \mathcal{U}_q, U \cap V \neq \emptyset\} \;\subseteq\; E(G_{\mathcal{U}}).$$

The ***pattern graph*** $H = H(P,\Psi)$ has vertex set P and set of edges

$$E(H) \;=\; \{pq \mid \|p-q\|_\infty \leq 2\Psi\} \;\subseteq\; \binom{P}{2}.$$

The use of the pattern graph is encoded in the following property: if $U \in \mathcal{U}_p$, $V \in \mathcal{U}_q$ and $U \cap V \neq \emptyset$, then $pq \in E(H)$. Indeed, if U and V intersect, then the union $U \cup V$ is contained in a square of side length 2Ψ, and thus the L_∞-distance between each $p \in P \cap U$ and $q \in P \cap V$ is at most 2Ψ.

The definition of $H(P,\Psi)$ implies that the edge set of $G_{\mathcal{U}}$ is the disjoint union of $E(\mathcal{U}_p,\mathcal{U}_q)$, over all $pq \in E(H)$, and the edge sets of the cliques $G_{\mathcal{U}_p}$, over all $p \in P$. Thus, whenever $pq \notin E(H)$, there are no edges in $E(\mathcal{U}_p,\mathcal{U}_q)$.

Let λ be the maximum degree of H. Note that $\lambda = O(\Psi^2)$. The value of λ is an upper bound on how many clusters $\mathcal{U}_q$ may interact with a single cluster $\mathcal{U}_p$. We will use λ as a parameter to decide how many objects from each $\mathcal{U}_p$ are kept. We start with a simple observation.

▶ **Lemma 8.** *There exists a maximum matching in $G_{\mathcal{U}}$ that, for all $pq \in E(H)$, contains at most one edge of $E(\mathcal{U}_p,\mathcal{U}_q)$.*

Of course we do not know which object from the cluster $\mathcal{U}_p$ will interact with another cluster $\mathcal{U}_q$. We will explain how to get a large enough subset of cluster $\mathcal{U}_p$.

For each $pq \in E(H)$, we construct a set $\mathcal{W}(p,q) \subseteq \mathcal{U}_p \cup \mathcal{U}_q$ as follows. First, we construct a matching $M = M(p,q)$ in $E(\mathcal{U}_p,\mathcal{U}_q)$ such that M has $2\lambda+1$ edges or M has fewer than $2\lambda+1$ edges and is maximal in $E(\mathcal{U}_p,\mathcal{U}_q)$. For example, such a matching can be constructed incrementally. If M has $2\lambda+1$ edges, we take $\mathcal{W}(p,q)$ to be the endpoints of M. Otherwise, for each endpoint $U \in \mathcal{U}_p$ (resp. $V \in \mathcal{U}_q$) of M, we place U (resp. V) and λ of its neighbors from $\mathcal{U}_q$ (resp. $\mathcal{U}_p$) into $\mathcal{W}(p,q)$. When U (resp. V) has fewer than λ neighbors, we place all its neighbors in $\mathcal{W}(p,q)$. This finishes the description of $\mathcal{W}(p,q)$; refer to Algorithm *Sparsify-one-edge* in the appendix (Figure 2) for pseudo-code.

▶ **Lemma 9.** *A maximum matching in*

$$\tilde{G} \;=\; \left(\bigcup_{pq \in E(H)} G_{\mathcal{W}(p,q)}\right) \cup \left(\bigcup_{p \in P} G_{\mathcal{U}_p}\right).$$

is a maximum matching in $G_{\mathcal{U}}$.

▶ **Lemma 10.** *The family of objects $\mathcal{W} = \cup_{pq \in E(H)} \mathcal{W}(p,q)$ has depth $O(\Psi^8)$.*

▶ **Theorem 11.** *Let $\mathcal{U}$ be a family of n geometric objects in the plane such that each object of $\mathcal{U}$ contains a square of side length 1 and is contained in a square of side length Ψ. Suppose that, for any $m \in \mathbb{N}$ and for any $p,q \in \mathbb{Z}^2$ with $|\mathcal{U}_p| + |\mathcal{U}_q| \leq m$, we can compute the sparsification $\mathcal{W}(p,q)$ as described above in time $T_{\text{spars}}(m)$, where $T_{\text{spars}}(m) = \Omega(m)$ is convex. In $O(\Psi^2 \cdot T_{\text{spars}}(n))$ time we can reduce the problem of finding a maximum matching in $G_{\mathcal{U}}$ to the problem of finding a maximum matching in $G_{\mathcal{W}}$ for some $\mathcal{W} \subseteq \mathcal{U}$ with maximum depth $O(\Psi^8)$.*

Algorithm *Sparsify-one-edge*
Input: p, q, $\mathcal{U}_p$ and $\mathcal{U}_q$
Output: $\mathcal{W}(p,q)$

$\mathcal{A}_p \leftarrow \mathcal{U}_p$
$\mathcal{A}_q \leftarrow \mathcal{U}_q$
($*$ compute matching M $*$)
$M \leftarrow \emptyset$
while $|M| < 2\lambda + 1$ **and** $\mathcal{A}_p \neq \emptyset$ **do**
    $U \leftarrow$ an arbitrary object of $\mathcal{A}_p$
    if U intersects some $V \in \mathcal{A}_q$ **then**
        $M \leftarrow M \cup \{UV\}$
        $\mathcal{A}_q \leftarrow \mathcal{A}_q \setminus \{V\}$
    $\mathcal{A}_p \leftarrow \mathcal{A}_p \setminus \{U\}$
($*$ end of computation of M $*$)
$\mathcal{W} \leftarrow \cup_{UV \in M}\{U, V\}$ ($*$ endpoints of M $*$)
if $|M| = 2\lambda + 1$ **then** ($*$ M large enough matching $*$)
    return $\mathcal{W}$
else ($*$ M maximal but small; add neighbors of $\mathcal{W}$ to the output $*$)
    $\mathcal{W}' \leftarrow \mathcal{W}$
    for $W \in \mathcal{W}$ **do**
        if $W \in \mathcal{U}_p$ **then**
            add to $\mathcal{W}'$ $\min\{\lambda, |E(\{W\}, \mathcal{U}_q)|\}$ elements of $\mathcal{U}_q$ intersecting W
        else ($*$ $W \in \mathcal{U}_p$ $*$)
            add to $\mathcal{W}'$ $\min\{\lambda, |E(\mathcal{U}_p, \{W\})|\}$ elements of $\mathcal{U}_p$ intersecting W
    return $\mathcal{W}'$

Figure 2 Algorithm *Sparsify-one-edge*

Our use of properties in the plane is very mild, and similar results hold in any space with constant dimension.

▶ **Theorem 12.** *Let $d \geq 3$ be a constant. Let $\mathcal{U}$ be a family of n geometric objects in $\mathbb{R}^d$ such that each object of $\mathcal{U}$ contains a cube of side length 1 and is contained in a cube of side length Ψ. Suppose that, for any $m \in \mathbb{N}$ and for any $p, q \in \mathbb{Z}^d$ with $|\mathcal{U}_p| + |\mathcal{U}_q| \leq m$, we can compute the sparsification $\mathcal{W}(p,q)$ as described above in time $T_{\text{spars}}(m)$, where $T_{\text{spars}}(m) = \Omega(m)$ is convex. In $O(\Psi^d \cdot T_{\text{spars}}(n))$ time we can reduce the problem of finding a maximum matching in $G_{\mathcal{U}}$ to the problem of finding a maximum matching in $G_{\mathcal{W}}$ for some $\mathcal{W} \subseteq \mathcal{U}$ with maximum depth $(1+\Psi)^{O(d)}$.*

As we mentioned in the introduction, for fat objects, bounded depth implies bounded density; see Har-Peled and Quanrud [13, Lemma 2.7]. If a convex object contains a cube of unit side length and is contained in a cube of side length Ψ, then it is $O(1/\Psi)$-fat; see van der Stappen et al. [26], where the parameter $1/\Psi$ goes under the name of thickness. Combining both results, one obtains that the relation between depth and density differs by a factor of

Ψ. For fixed shapes, they depth and density differ by a constant factor.

5 Efficient sparsification

Now, we implement Algorithm *Sparsify-one-edge* (Figure 2) efficiently. In particular, we must perform the test in line 7 and find the neighbors in line 19 (and the symmetric case in line 21). The shape of the geometric objects becomes relevant for this. First, we note that it suffices to obtain an efficient semi-dynamic data structure for intersection queries.

▶ **Lemma 13.** *Suppose there is a data structure with the following properties: for any $m \in \mathbb{N}$ and for any $p, q \in \mathbb{Z}^2$ with $|\mathcal{U}_p| + |\mathcal{U}_q| \leq m$, we can maintain a set $\mathcal{A}_q \subseteq \mathcal{U}_q$ under deletions so that, for any query $U \in \mathcal{U}_p$, we either find some $V \in \mathcal{A}_q$ with $U \cap V \neq \emptyset$ or correctly report that no such V exists. Let $T_{\text{con}}(m)$ be the time to construct the data structure, $T_{\text{que}}(m)$ an upper bound on the amortized query time, and $T_{\text{del}}(m)$ be an upper bound on the amortized deletion time. Then, the running time of Algorithm Sparsify-one-edge (Figure 2) for the input $(p, q, \mathcal{U}_p, \mathcal{U}_q)$ is $T_{\text{sparse}}(m) = O(T_{\text{con}}(m) + mT_{\text{que}}(m) + \lambda^2 T_{\text{del}}(m))$.*

5.1 Disks in the plane

When $\mathcal{U}$ consists of disks in the plane, we can use the data structure of Kaplan et al. [15] to sparsify an edge of the pattern graph. This leads to the following.

▶ **Proposition 14.** *Consider a family $\mathcal{U}$ of n disks in the plane with radii in $[1, \Psi]$. In $O(\Psi^6 n \log^{11} n)$ expected time, we can reduce the problem of finding a maximum matching in $G_{\mathcal{U}}$ to the problem of finding a maximum matching in $G_{\mathcal{W}}$ for some subfamily $\mathcal{W} \subseteq \mathcal{U}$ of disks with maximum depth $O(\Psi^8)$.*

Possibly, the method can be extended to homothets of a single object. For this one should consider the surfaces defined by weighted distances in the approach of Kaplan et al. [15].

Since the depth and the density of a family of disks are linearly related, Proposition 14 and Theorem 7 with $\rho = O(\Psi^8)$ imply the following.

▶ **Theorem 15.** *Consider a family $\mathcal{U}$ of n disks in the plane with radii in the interval $[1, \Psi]$. In $O(\Psi^6 n \log^{11} n + \Psi^{12\omega} n^{\omega/2})$ expected time, we can compute a matching in $G_{\mathcal{U}}$ that, with high probability, is maximum.*

5.2 Translates of a fixed convex shape in the plane

Now, suppose $\mathcal{U}$ consists of translates of a single convex object with non-empty interior in the plane. With an affine transformation, we ensure that the object is *fat*: the radii of the minimum enclosing disk and of the maximum enclosed disk are within a constant factor. Such a transformation is standard; e.g., [1, Lemma 3.2]. Thus, we may assume that $\Psi = O(1)$. We start with a standard lemma.

▶ **Lemma 16.** *Let $\mathcal{U}$ be a family of n translates of a convex object in the plane that are pierced by a given point q. The union of $\mathcal{U}$ can be computed in $O(n \log n)$ time.*

We will use the following lemma to "simulate" deletions. For this, we will keep a half-infinite interval of indices that contains the elements that are "deleted".

▶ **Lemma 17.** *Let $\mathcal{U} = \{U_1, \ldots U_n\}$ be a family of n translates of a convex object in the plane that are pierced by a given point q. In $O(n \log^2 n)$ time, we can construct a data structure for*

the following queries: given $x \in \mathbb{R}^2$ and a value $a \in \{1, \ldots, n\}$, find the smallest $i \geq a$ such that U_i contains x, or correctly report that x does not belong to $U_a \cup \cdots \cup U_n$. The query time is $O(\log^2 n)$.

▶ **Lemma 18.** *Let $\mathcal{U}_q = \{V_1, \ldots V_n\}$ be a family of n translates of a convex object in the plane that are pierced by a given point q. Let U_0 be a convex object. In $O(n \log^2 n)$ time, we can construct a data structure for the following type of queries: given a translate U of U_0 and a value a, find the smallest $i \geq a$ such that U intersects V_i, or correctly report that U does not intersect $V_a \cup \cdots \cup V_n$. Each query can be answered in $O(\log^2 n)$ time.*

Lemma 18 can be used to make queries and simulate deletions.

▶ **Proposition 19.** *Consider a family $\mathcal{U}$ of n translates of a convex object with non-empty interior in the plane. In $O(n \log^2 n)$ time, we can reduce the problem of finding a maximum matching in $G_{\mathcal{U}}$ to the problem of finding a maximum matching in $G_{\mathcal{W}}$ for some subfamily $\mathcal{W} \subseteq \mathcal{U}$ with maximum depth $O(1)$.*

Combining Proposition 19 and Theorem 7 we obtain the following.

▶ **Theorem 20.** *Consider a family $\mathcal{U}$ of translates of a convex object with non-empty interior in the plane. In $O(n^{\omega/2})$ time we can find matching in $G_{\mathcal{U}}$ that, with high probability, is maximum.*

If $\mathcal{U}$ consists of unit disks,the sparsification can be done slightly faster using a semi-dynamic data structure by Efrat, Itai, and Katz [9], which has $O(T_{\text{con}}(m)) = O(m \log m)$, and $O(T_{\text{op}}(m)) = O(\log m)$. However the current bottleneck is the computation of the maximum matching *after* the sparsification. Thus, improving the sparsification in the particular case of unit disks does not lead to an improved final algorithm.

Proposition 19 and Theorem 20 also holds if we have translations of $O(1)$ different convex objects (with nonempty interiors). Indeed, the data structure of Lemma 18 can be made for each pair of different convex shapes. In this case, the constant Ψ depends on the shapes, namely the size of the largest square that we can place inside each of the convex shapes and the size of the smallest square that can be used to cover each of the convex shapes. Also, the relation between the depth and the density depends on the shapes. However, for a fixed set of $O(1)$ shapes, both values are constants that depend on the shapes.

▶ **Theorem 21.** *Consider a family $\mathcal{U}$ of translates of a constant number of different convex objects in the plane with non-empty interiors. In $O(n^{\omega/2})$ time we can find matching in $G_{\mathcal{U}}$ that, with high probability, is maximum.*

5.3 Axis-parallel objects

A ***box*** is the Cartesian product of intervals. Combining standard data structures for orthogonal range searching [6, Sections 5.4 and 10.3] one obtains the following results.

▶ **Proposition 22.** *Let $d \geq 2$ be an integral constant. Consider a family $\mathcal{U}$ of n boxes in $\mathbb{R}^d$ such that each box of $\mathcal{U}$ contains a cube of side length 1 and is contained in a cube of side length Ψ. In $O(\Psi^d \cdot n \operatorname{polylog} n)$ time we can reduce the problem of finding a maximum matching in $G_{\mathcal{U}}$ to the problem of finding a maximum matching in $G_{\mathcal{W}}$, for some $\mathcal{W} \subseteq \mathcal{U}$ with maximum depth $(1 + \Psi)^{O(d)}$.*

For $d = 2$, we can combine Theorem 7 and Proposition 22. Since we have assumed $\omega > 2$, the $O(n \operatorname{polylog} n)$ term is asymptotically smaller than $O(n^{\omega/2})$, and we obtain the following.

▶ **Theorem 23.** *Given a family $\mathcal{U}$ of n boxes in $\mathbb{R}^2$ such that each object of $\mathcal{U}$ contains a square of side length 1 and is contained in a square of side length Ψ, we can compute in $(1+\Psi)^{O(1)}n^{\omega/2}$ time a matching in $G_{\mathcal{U}}$ that, with high probability, is a maximum matching.*

Consider now the case $d \geq 3$. The set $\mathcal{W}$ that we obtain from Proposition 22 has depth and density $\rho = (1+\Psi)^{O(d)}$, and therefore the graph $G_{\mathcal{W}}$ has $O(\rho n)$ edges; see Lemma 1. We can thus use the algorithm of Mądry [19], which takes $\tilde{O}(|E(G_{\mathcal{W}})|^{10/7})) = \tilde{O}((1+\Psi)^{O(d)}n^{10/7})$ time. We summarize.

▶ **Corollary 24.** *Let $d \geq 3$ be an integral constant. Given a family $\mathcal{U}$ of n boxes in $\mathbb{R}^d$ such that each object of $\mathcal{U}$ contains a cube of side length 1 and is contained in a cube of side length Ψ, we can compute in $\tilde{O}((1+\Psi)^{O(d)}n^{10/7})$ time a matching in $G_{\mathcal{U}}$ that, with high probability, is a maximum matching.*

5.4 Congruent balls in $d \geq 3$ dimensions

Consider now the case of congruent balls in $\mathbb{R}^d$, for constant $d \geq 3$. Note that $\lambda = O(1)$ in this case. We use the dynamic data structure by Agarwal and Matoušek [2] for the sparsification. For each m with $n \leq m \leq n^{\lceil d/2 \rceil}$, the data structure maintains n points in $\mathbb{R}^d$, answers $O(n)$ queries for closest point and supports $O(\lambda^2)$ updates in

$$O\left(m^{1+\varepsilon} + \lambda^2 \frac{m^{1+\varepsilon}}{n} + n \cdot \frac{n \log^3 n}{m^{1/\lceil d/2 \rceil}}\right)$$

time. Here $\varepsilon > 0$, is an arbitrary constant whose choice affects to the constants hidden in the O-notation. For $d \in \{3, 4\}$, this running time is

$$O\left(m^{1+\varepsilon} + \lambda^2 \frac{m^{1+\varepsilon}}{n} + n \cdot \frac{n \log^3 n}{m^{1/2}}\right).$$

Setting $m = n^{4/3}$, we get a running time of $O(n^{4/3+\varepsilon} + \lambda^2 n^{1/3+\varepsilon}) = O(n^{4/3+\varepsilon})$ to handle $O(n)$ queries and $O(\lambda^2) = O(1)$ updates. Using this in Lemma 13 and Theorem 12, we get the following result

▶ **Proposition 25.** *Consider a family $\mathcal{U}$ of n unit balls objects in $\mathbb{R}^d$, for $d \in \{3, 4\}$. In $O(n^{4/3+\varepsilon})$ time, we can reduce the problem of finding a maximum matching in $G_{\mathcal{U}}$ to the problem of finding a maximum matching in $G_{\mathcal{W}}$ for some $\mathcal{W} \subseteq \mathcal{U}$ with maximum depth $O(1)$.*

For the resulting set $\mathcal{W}$ with depth $O(1)$, it is better to use the algorithm of Mądry [19] for sparse graphs. Note that $G_{\mathcal{W}}$ is sparse, and thus has $O(n)$ edges. Therefore, a maximum matching in $G_{\mathcal{W}}$ can be computed in $O(n^{10/7})$ time. In summary, we spend $O(n^{4/3+\varepsilon})$ for the sparsification and $O(n^{10/7})$ for computing the matching in the sparsified setting.

For $d > 4$, we set $m = n^{\frac{2\lceil d/2 \rceil}{1+\lceil d/2 \rceil}}$. The running time for the sparsification is then $O(n^{\frac{2\lceil d/2 \rceil}{1+\lceil d/2 \rceil}+\varepsilon})$. For each constant d, the resulting instance $G_{\mathcal{W}}$ has $O(n)$ edges. For $d = 5, 6$, the running time of the sparsification is $O(n^{3/2+\varepsilon})$. However, after the sparsification, we have a graph with $O(n)$ edges, and we can use the algorithm of Micali and Vazirani [20], which takes $O(n^{3/2})$ time. Thus, for $d \geq 5$, the running time is dominated by the sparsification.

▶ **Theorem 26.** *Let $d \geq 3$ be a constant. Consider a family $\mathcal{U}$ of congruent balls in $\mathbb{R}^d$. For $d = 3, 4$, we can find in $O(n^{10/7})$ time a maximum matching in $G_{\mathcal{U}}$. For $d \geq 5$, we can find in $O(n^{\frac{2\lceil d/2 \rceil}{1+\lceil d/2 \rceil}+\varepsilon})$ time a maximum matching in $G_{\mathcal{U}}$, for each $\varepsilon > 0$.*

6 Conclusion

We have proposed the density of a geometric intersection graph as a parameter for the maximum matching problem, and we showed that it can be fruitful in obtaining efficient matching algorithms. Then, we presented a sparsification method that lets us reduce the general problem to the case of bounded density for several interesting classes of geometric intersection graphs. In our sparsification method, we did not attempt to optimize the dependency on the radius ratio Ψ. It may well be that this can be improved by using more advanced grid-based techniques. Furthermore, our sparsification needs the complete intersection graph and does not apply to the bipartite setting. Here, we do not know of a method to reduce the general case to bounded density. In general, the complexity of the matching problem is wide open. To the best of our knowledge, there are no (even weak) superlinear lower bounds for the (static) matching problem in general graphs.

References

1 Pankaj. K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004. doi:http://doi.acm.org/10.1145/1008731.1008736.

2 Pankaj K. Agarwal and Jiří Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993. doi:10.1137/0222051.

3 Jochen Alber and Jirí Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004.

4 Noga Alon and Raphael Yuster. Matrix sparsification and nested dissection over arbitrary fields. *J. ACM*, 60(4):25:1–25:18, 2013. URL: http://doi.acm.org/10.1145/2505989.

5 Édouard Bonnet, Sergio Cabello, and Wolfgang Mulzer. Maximum matchings in geometric intersection graphs. *CoRR*, abs/1910.02123, 2019. URL: http://arxiv.org/abs/1910.02123, arXiv:1910.02123.

6 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: http://www.worldcat.org/oclc/227584184.

7 Mark de Berg, A. Frank van der Stappen, Jules Vleugels, and Matthew J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002. URL: https://doi.org/10.1007/s00453-002-0961-x.

8 Hristo Djidjev and Shankar M. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Inf.*, 34(3):231–243, 1997.

9 Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001. URL: https://doi.org/10.1007/s00453-001-0016-8.

10 David Eppstein, Gary L. Miller, and Shang-Hua Teng. A deterministic linear time algorithm for geometric separators and its applications. *Fundam. Inform.*, 22(4):309–329, 1995.

11 J. Gao and L. Guibas. Geometric algorithms for sensor networks. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 370(1958):27–51, 2011. URL: http://dx.doi.org/10.1098/rsta.2011.0215.

12 John R. Gilbert and Robert E. Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50(4):377–404, 1986. URL: https://doi.org/10.1007/BF01396660.

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

14 M. L. Huson and A. Sen. Broadcast scheduling algorithms for radio networks. In *IEEE MILCOM '95*, volume 2, pages 647–651 vol.2, 1995. URL: http://dx.doi.org/10.1109/MILCOM.1995.483546.

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

16 Santosh Kumar, Ten H. Lai, and Anish Arora. Barrier coverage with wireless sensors. *Wireless Networks*, 13(6):817–834, 2007.

17 Richard J. Lipton, Donald J. Rose, and Robert E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16(2):346–358, 1979. URL: https://doi.org/10.1137/0716027.

18 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.

19 Aleksander Mądry. Navigating central path with electrical flows: From flows to matchings, and back. In *Proceedings of the 54th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2013*, pages 253–262, 2013. URL: https://doi.org/10.1109/FOCS.2013.35.

20 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}\cdot|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science, FOCS 1980*, pages 17–27, 1980. URL: https://doi.org/10.1109/SFCS.1980.12.

21 Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997.

22 Marcin Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. In *Proceedings of the 45th Symposium on Foundations of Computer Science, FOCS 2004*, pages 248–255, 2004. URL: https://doi.org/10.1109/FOCS.2004.40.

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

24 Michael O. Rabin and Vijay V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithms*, 10(4):557–567, 1989. URL: https://doi.org/10.1016/0196-6774(89)90005-9.

25 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems and applications. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FOCS 1998*, pages 232–243, 1998.

26 A. Frank van der Stappen, Dan Halperin, and Mark H. Overmars. The complexity of the free space for a robot moving amidst fat obstacles. *Comput. Geom.*, 3:353–373, 1993. doi:10.1016/0925-7721(93)90007-S.

27 Vijay V. Vazirani. A simplification of the MV matching algorithm and its proof. arVix:1210.4594, 2012. URL: http://arxiv.org/abs/1210.4594.

28 Raphael Yuster and Uri Zwick. Maximum matching in graphs with an excluded minor. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, pages 108–117, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283396.

29 F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Elsevier/Morgan-Kaufmann, 2004.