Algorithms and Turing Kernels for Detecting and Counting Small Patterns in Unit Disk Graphs

Jesper Nederlof and Krisztina Szilágyi(✉)

Utrecht University, Utrecht, The Netherlands
{j.nederlof,k.szilagyi}@uu.nl

Abstract. In this paper we investigate the parameterized complexity of the task of counting and detecting occurrences of small patterns in unit disk graphs: Given an n-vertex unit disk graph G with an embedding of ply p (that is, the graph is represented as intersection graph with closed disks of unit size, and each point is contained in at most p disks) and a k-vertex unit disk graph P, count the number of (induced) copies of P in G.

For general patterns P, we give an $2^{O(pk/\log k)}n^{O(1)}$ time algorithm for counting pattern occurrences. We show this is tight, even for ply $p = 2$ and $k = n$: any $2^{o(n/\log n)}n^{O(1)}$ time algorithm violates the Exponential Time Hypothesis (ETH).

For most natural classes of patterns, such as connected graphs and independent sets we present the following results: First, we give an $(pk)^{O(\sqrt{pk})}n^{O(1)}$ time algorithm, which is nearly tight under the ETH for bounded ply and many patterns. Second, for $p = k^{O(1)}$ we provide a Turing kernelization (i.e. we give a polynomial time preprocessing algorithm to reduce the instance size to $k^{O(1)}$).

Our approach combines previous tools developed for planar subgraph isomorphism such as 'efficient inclusion-exclusion' from [Nederlof STOC'20], and 'isomorphisms checks' from [Bodlaender et al. ICALP'16] with a different separator hierarchy and a new bound on the number of non-isomorphic separations of small order tailored for unit disk graphs.

Keywords: Unit disk graphs · Subgraph isomorphism · Parameterized complexity

1 Introduction

A well-studied theme within the complexity of computational problems on graphs is how much structure within inputs allows faster algorithms. One of the most active research directions herein is to assume that input graphs are

Supported by the project CRACKNP that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 853234).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
H. Fernau et al. (Eds.): SOFSEM 2024, LNCS 14519, pp. 413–426, 2024.
https://doi.org/10.1007/978-3-031-52113-3_29

geometrically structured. The (arguably) two most natural and commonly studied variants of this are to assume that the graph can be drawn on $\mathbb{R}^2$ without crossings (i.e., it is planar) or it is the intersection graph of simple geometric objects. While this last assumption can amount to a variety of different models, a canonical and most simple model is that of *unit disk graphs*: Each vertex of the graph is represented by a disk of unit size and two vertices are adjacent if and only if the two associated disks intersect.

The computational complexity of problems on planar graphs has been a very fruitful subject of study: It led to the development of powerful tools such as *Bakers layering technique* and bidimensionality that gave rise to efficient approximation schemes and fast (parameterized) sub-exponential time algorithms for many NP-complete problems. One interesting example of such an NP-complete problem is *(induced) subgraph isomorphism*: Given a k-vertex pattern P and n-vertex host graph G, detect or count the number of (induced) copies of P inside G, denoted with $\mathrm{sub}(P, G)$ (respectively, $\mathrm{ind}(P, G)$). Here we think of k as being much smaller than n, and therefore it is very interesting to obtain running times that are only exponential in k (i.e. *Fixed Parameter Tractable time*). This problem is especially appealing since it generalizes many natural NP-complete problems (such as Independent Set, Longest Path and Hamiltonian Cycle) in a natural way, but its generality poses significant challenges for the bidimensionality theory: It does not give sub-exponential time algorithms for this problem.

Only recently, it was shown in a combination of papers ([4,7] and subsequently [13]) that, on planar graphs, subgraph isomorphism can be solved in $2^{O(k/\log k)}$ time[1] for general patterns, and in $2^{\tilde{O}(\sqrt{k})}n^{O(1)}$ time for many natural pattern classes, complementing the lower bound of $2^{o(n/\log n)}$ time from [4] based on the Exponential Time Hypothesis (ETH). It was shown in [13] that (induced) pattern occurrences can even be *counted* in sub-exponential parameterized (i.e. $2^{o(k)}n^{O(1)}$) time.

Unfortunately, most of these methods do not immediately work for (induced) subgraph isomorphism on geometric intersection graphs: Even unit disk graphs of bounded ply[2] are not H-minor free for any graph H (which significantly undermines the bidimensionality theory approach), and unit disk graphs of unbounded ply can even have arbitrary large cliques. This hardness is inherent to the graph class: Independent Set is $W[1]$-hard on unit disk graphs and, unless ETH fails, and it has no $f(k)n^{o(\sqrt{k})}$-time algorithm for any computable function f [6, Theorem 14.34]. On the positive side, it was shown in [14] that for bounded expansion graphs and fixed patterns, the subgraph isomorphism problem can be solved in linear time, which implies that subgraph isomorphism is fixed parameter tractable on unit disk graphs; however, their method relies on Courcelle's theorem and hence the dependence of k in their running time is very large and far from optimal.

[1] We let $\tilde{O}()$ omit polylogarithmic factors in its argument.

[2] The ply of an embedded unit disk graph is defined as the maximum number of times any point of the plane is contained in a disk.

Therefore, a popular research topic has been to design such fast (parameterized) sub-exponential time algorithms for specific problems such as Independent Set, Hamiltonian Cycle and Steiner Tree [2,3,10,16].

In this paper we continue this research line by studying the complexity of the decision and counting version of (induced) subgraph isomorphism. While some general methods such as contraction decompositions [15] and pattern covering [12] were already designed for graph classes that include (bounded ply) unit disk graphs, the fine-grained complexity of the subgraph problem itself restricted to unit disk graphs has not been studied and is still far from being understood.

Our Results. To facilitate the formal statements of our results, we need the following definitions: Given two graphs P and G, we define

$$\text{ind}(P,G) = \{f : V(P) \to V(G) : f \text{ is injective}, uv \in E(P) \Leftrightarrow f(u)f(v) \in E(G)\}$$
$$\text{sub}(P,G) = \{f : V(P) \to V(G) : f \text{ is injective}, uv \in E(P) \Rightarrow f(u)f(v) \in E(G)\}$$

Our main theorem reads as follows:

Theorem 1. *There is an algorithm that takes as input unit disk graphs P and G on k vertices and n vertices respectively, together with disk embeddings of ply p. It outputs $|\text{sub}(P,G)|$ and $|\text{ind}(P,G)|$ in $(pk)^{O(\sqrt{pk})}\sigma_{O(\sqrt{pk})}(P)^2 n^{O(1)}$ time.*

In this theorem, the parameter σ_s is a somewhat technical parameter of the pattern graph P that is defined as follows:

Definition 1 ([13]). *Given a graph P, we say that (A,B) is a* separation *of P of order s if $A \cup B = V(P)$, $|A \cap B| = s$ and there are no edges between $A \setminus B$ and $B \setminus A$. We say that (A,B) and (C,D) are* isomorphic separations *of P if there is a bijection $f : V(P) \to V(P)$ such that*

- *For any $u,v \in V(P)$, $uv \in E(P) \Leftrightarrow f(u)f(v) \in E(P)$*
- *$f(A) = C$, $f(B) = D$,*
- *For any $u \in A \cap B$, $f(u) = u$*

We denote by $\mathcal{S}_s(P)$ a maximal set of pairwise non-isomorphic separations of P of order at most s. We define $\sigma_s(P) = |\mathcal{S}_s(P)|$ as the number of non-isomorphic separations of P of order at most s.

Note that $\sigma_s(P) \geq \binom{k}{s}s!$. For ply $p = O(1)$ and many natural classes of patterns such as independent sets, cycles, or grids, it is easy to see that $\sigma_{O(\sqrt{k})}(P)$ is at most $2^{\tilde{O}(\sqrt{k})}$ and therefore Theorem 1 gives a $2^{\tilde{O}(\sqrt{k})}n^{O(1)}$ time algorithm for computing $|\text{sub}(P,G)|$ and $|\text{ind}(P,G)|$. We also give the following new non-trivial bounds on $\sigma_s(P)$ whenever P is a general (connected) unit disk graph[3]:

Theorem 2. *Let P be a k-vertex unit disk graph with given embedding of ply p, and s be an integer. Then: (a) $\sigma_s(P)$ is at most $2^{O(s\log k + pk/\log k)}$, and (b) If P is connected, then $\sigma_s(P) \leq 2^{O(s\log k)}$.*

[3] The proof of Theorem 2 can be found in the full version of the paper.

Using Theorem 1, this allows us to conclude the following result.

Corollary 1. *There is an algorithm that takes as input a k-vertex unit disk graph P and an n-vertex unit disk graph G, together with their unit disk embeddings of ply p, and outputs* $|\mathrm{sub}(P,G)|$ *and* $|\mathrm{ind}(P,G)|$ *in* $2^{O(pk/\log k)}n^{O(1)}$ *time.*

We show that this cannot be significantly improved even if $k = n$:

Theorem 3. *Assuming ETH, there is no algorithm to determine if* $\mathrm{sub}(P,G)$ *(*$\mathrm{ind}(P,G)$ *respectively) is nonempty for given n-vertex unit disk graphs P and G in* $2^{o(n/\log n)}$ *time, even when P and G have a given embedding of ply* 2.

Note that the ply of G is 1 if and only if G is an independent set so the assumption on the ply in the above statement is necessary. To our knowledge, this is the first lower bound based on the ETH excluding $2^{O(\sqrt{n})}$ time algorithms for problems on unit disks graphs (of bounded ply), in contrast to previous bounds that only exclude $2^{o(\sqrt{n})}$ time algorithms.

Clearly, a unit disk graph G of ply p has a clique of size p, i.e. its clique number $\omega(G)$ is at least p. On the other hand, it was shown in [8] that $p \geq \omega(G)/5$. In other words, parameterizations by ply and clique number are equivalent up to a constant factor.

1.1 Our Techniques

Our approach heavily builds on the previous works [4,7,13]: Theorem 1 is proved by using the dynamic programming technique from [4] that stores representatives of non-isomorphic separations to get the runtime dependence down from $2^{O(k)}$ to $\sigma_{O(\sqrt{pk})}(P)$, and the *Efficient Inclusion-Exclusion* technique from [13] to solve counting problems (on top of decision problems) as well.[4] We combine these techniques with a divide and conquer strategy that divides the unit disk graph in smaller graphs using horizontal and vertical lines as separators. As a first step in our proof, we give an interesting *Turing kernelization* for the counting problems that uses efficient inclusion-exclusion. Theorem 2 uses a proof strategy from [4] combined with a bound from [5] on the number of non-isomorphic unit disk graphs. Theorem 3 builds on a reduction from [4], although several alterations are needed to ensure the graph is a unit disk graph of bounded ply.

Organization. In Sect. 2 we provide additional notation and some preliminary lemmas. In Sect. 3 we provide a Turing kernel. In Sect. 4 we build on Sect. 3 to provide the proof of Theorem 1. In Sect. 5 we give an outline of the proof of Theorem 3. We finish with some concluding remarks in Sect. 6. The proofs that are omitted due to space restrictions (indicated with †) are provided in the full version on arXiv.

[4] Similar to what was discussed in [13], this technique seems to be needed even for simple special cases of Theorem 1 such as counting independent sets on subgraphs of (subdivided) grids.

2 Preliminaries

Notation. Given a graph G and a subset A of its vertices, we define $G[A]$ as the subgraph of G induced by A. Given a unit disk graph G, we say that G has *ply* p if there is an embedding of G such that every point in the plane is contained in at most p disks of G. Let P and G be unit disk graphs and let $|V(G)| = n$, $|V(P)| = k$.

We denote all vectors by bold letters, the all ones vector by $\mathbf{1}$ and the all zeros vector by $\mathbf{0}$. We use the Iverson bracket notation: for a statement P, we define $[P] = 1$ if P is true and $[P] = 0$ if P is false. We define $[k] = \{1, \dots, k\}$. Given a function $f : A \to B$ and $C \subseteq A$, we define the restriction of f to C as $f|_C : C \to B$, $f|_C(c) = f(c)$ for all $c \in C$. If $g = f|_C$ for some C then we say that f *extends* g.

Definition 2. *For integers b, h, we say a unit disk graph G of ply p can be* drawn in a $(b \times h)$-box *with ply p if it has an embedding of ply p as unit disk graph in a $b \times h$ rectangle.*

Throughout this paper, we assume that the sides of the box are axis parallel, and the lower left corner is at $(0, 0)$. We assume that if a graph G can be drawn in a $(b \times h)$-box then we are given such an embedding.

Lemma 1 (†). *Given a unit disk graph G with a drawing in a $(b \times h)$-box with ply p, one can construct in polynomial time a path decomposition of G of width $4(\min\{b, h\} + 1)p$.*

Lemma 2 (Theorem 6.1 from [5]). *Let a non-decreasing bound $b = b(n)$ be given, and let $\mathcal{U}_n$ denote the set of unlabeled unit disk graphs on n vertices with maximum clique size at most b. Then $|\mathcal{U}_n| \leq 2^{12(b+1)n}$.*

Subroutines. The following lemma can be shown with standard dynamic programming over tree decompositions:

Lemma 3 (†). *Given P, G, a subset $P' \subseteq V(P)$, a path decomposition of G of width t, and a function $f : P' \to V(G)$, we can count $|\{g \in \mathrm{sub}(P, G) : g \text{ extends } f\}|$ and $|\{g \in \mathrm{ind}(P, G) : g \text{ extends } f\}|$ in time $\sigma_t(P) t^t n^{O(1)}$.*

The following lemma simply states that a long product of matrices can be evaluated quickly, but is nevertheless useful in a subroutine in the 'efficient inclusion-exclusion' technique.

Lemma 4 ([13]). *Given a set A, an integer h and a value $T[x, x'] \in \mathbb{Z}$ for every $x, x' \in A$, the value*

$$\sum_{x_1, \dots, x_h \in A} \prod_{i=1}^{h-1} T[x_i, x_{i+1}] \tag{1}$$

can be computed in $O(h|A|^2)$ time.

Non-isomorphic Separations. In this paper we will work with non-isomorphic separations of small order, as defined in Definition 1. For separations $(C_1, D_1), (C_2, D_2) \in \mathcal{S}_s(P)$, we define

$$\mu((C_1, D_1), (C_2, D_2)) = |\{(C, D) :\ (C, D) \text{ is a separation of } P \text{ such that}$$
$$C \subseteq C_2 \text{ and } (C, D) \text{ isomorphic to } (C_1, D_1)\}|$$

Lemma 5 (†). *Given a graph P, one can compute $\mathcal{S}_s(P)$ and for each pair of separations $(C_1, D_1), (C_2, D_2) \in \mathcal{S}_s(P)$ the multiplicity $\mu((C_1, D_1), (C_2, D_2))$ in time $\sigma_s(P)n^{O(1)}$.*

3 Turing Kernel

We will now present a preprocessing algorithm for computing $|\text{sub}(P, G)|$ (the algorithm for $|\text{ind}(P, G)|$ is analogous) that allows us to assume that G can be drawn in a $(k \times k)$-box with ply p, i.e. that $|V(G)| \leq k^2 p$. This can be seen as a polynomial Turing kernel in case p and $\sigma_0(P)$ are polynomial in k. A *Turing kernel* of size f is an algorithm that solves the given problem in polynomial time, when given access to an oracle that solves instances of size at most $f(k)$ in a single step.

Lemma 6 describes how to reduce the width of G (and analogously the height of G). To prove it, we use the shifting technique. This general technique was first used by Baker [1] for covering and packing problems on planar graphs and by Hochbaum and Maass [9] for geometric problems stemming from VLSI design and image processing.

Intuitively, we draw the graph on a grid, and delete all the disks that intersect certain columns of the grid. After doing that, the remaining graph will consist of several small disconnected "building blocks". Each connected component of the pattern will be fully contained in one of the blocks, and since the blocks are small we can use the oracle to count the number of these occurrences. We take advantage of the fact that we can group together connected components that are isomorphic. We use Lemma 6 twice, to reduce the width and height of G to k.

Lemma 6. *Suppose we have access to an oracle that computes $|\text{sub}(P, G)|$ in constant time, where the host graph G can be drawn in a $(O(k) \times O(k))$-box with ply p. Then we can compute $|\text{sub}(P, G')|$ for host graphs G' that can be drawn in a box of height k with ply p in time $n \cdot \text{poly}(k) \cdot \sigma_0(P)^2$.*

Proof. For $i \in \{0, \ldots, k\}$ let $C_i = \{(x, y) \in \mathbb{R}^2 :\ x \equiv i \pmod{k+1}\}$. Informally, we draw a grid and select every $(k+1)$th vertical gridline. Let P_i be the set of all subgraphs of G that are isomorphic to P and are disjoint from C_i.

Note that every subgraph Q of G that is isomorphic to P is fully contained in at least one P_i. Indeed, every disk in Q can intersect at most one vertical grid line, so Q is disjoint from C_i for at least one value of i.

By the inclusion-exclusion principle, $\left|\bigcup_{i=0}^{k} P_i\right|$ equals

$$\sum_{\emptyset \subset C \subseteq \{0,\dots,k\}} (-1)^{|C|} \left| \bigcap_{i \in C} P_i \right| = \sum_{\ell=1}^{k+1} (-1)^{\ell} \sum_{0 \le c_1 < \cdots < c_\ell \le k} \left| \bigcap_{j \in \{c_1,\dots,c_\ell\}} P_j \right|. \quad (2)$$

Let us show how we can compute $|\cap_{j\in\{c_1,\dots,c_\ell\}} P_j|$ quickly. For $a, b \in \{0,\dots,k\}$, we define $\mathcal{B}[a,b]$ as the subgraph of G contained by (open) stripes bounded by C_a and C_b. Formally, define $B[a,b]$ to be

$$\begin{cases} \{(x,y) \in \mathbb{R}^2 : (\exists t \in \mathbb{N}_0)\; a + (k+1)t < x < b + (k+1)t\}, & \text{if } a \le b, \\ \{(x,y) \in \mathbb{R}^2 : (\exists t \in \mathbb{N}_0)\; a + (k+1)(t-1) < x < b + (k+1)t\}, & \text{if } a > b. \end{cases}$$

We define $\mathcal{B}[a,b]$ as the induced subgraph of G such that all its disks are fully contained in $B[a,b]$. These sets are our "building blocks": after deleting $C_{c_1}, \dots C_{c_\ell}$, the remaining graph is $\cup_{\alpha=1}^{\ell} \mathcal{B}[c_\alpha, c_{\alpha+1}]$, where we define $c_{\ell+1} = c_1$.

Let t be the number of non-isomorphic connected components of P and let $\mathcal{C}_0(P) = \{\mathcal{P}_1, \dots, \mathcal{P}_t\}$ be the set of representatives of all isomorphism classes of connected components of P. We can encode P as vector $\mathbf{p} = (p_1, \dots, p_t)$, where p_i is the size of the isomorphism class of $\mathcal{P}_i$.

Let $U = \{0,\dots,p_1\} \times \cdots \times \{0,\dots,p_t\}$. For a t-dimensional vector $(v_1,\dots,v_t) \in U$ we define $P[(v_1,\dots,v_t)]$ as the subgraph of P that contains v_i copies of $\mathcal{P}_i$.

We would like to count in how many ways can we distribute the connected components of P to the building blocks. Equivalently, we can count the number of ways to assign a vector $\mathbf{v}^\alpha \in U$ to each block $\mathcal{B}[c_\alpha, c_{\alpha+1}]$ such that $\sum \mathbf{v}^\alpha = \mathbf{p}$.

Thus we have

$$\left| \bigcap_{j \in \{c_1,\dots,c_\ell\}} P_j \right| = \sum_{\mathbf{v}^1 + \cdots + \mathbf{v}^\ell = \mathbf{p}} \prod_{\alpha=1}^{\ell} |\mathrm{sub}(P[\mathbf{v}^\alpha], \mathcal{B}[c_\alpha, c_{\alpha+1}])|. \quad (3)$$

Note that $|U| = (p_1+1)\cdots(p_t+1) = \sigma_0(P)$: indeed, every vector $\mathbf{u} \in U$ corresponds to a unique separation $(V(P'), V(P-P'))$ of order 0, where P' consists of c_i copies of $\mathcal{P}_i$. Combining (2) and (3), we get that

$$\left| \bigcup_{i=0}^{k} P_i \right| = \sum_{\ell=1}^{k} (-1)^{\ell} T_\ell,$$

where

$$T_\ell = \sum_{\substack{0 \le c_1 < \cdots < c_\ell \le k \\ \mathbf{v}^1 + \cdots + \mathbf{v}^\ell = \mathbf{p}}} \prod_{\alpha=1}^{\ell} |\mathrm{sub}(P[\mathbf{v}^\alpha], \mathcal{B}[c_\alpha, c_{\alpha+1}])|. \quad (4)$$

Suppose for now that we have computed $|\mathrm{sub}(P[\mathbf{v}^\alpha], \mathcal{B}[a,b])|$ for all $a, b \in \{0,\dots,k\}$, $\mathbf{v}^\alpha \in U$, and that we want to compute T_ℓ quickly.

To apply Lemma 4, we have to rewrite the sum (4) in such a way that the variables are pairwise independent. We replace the condition $c_i < c_{i+1}$ by multiplying with $[c_i < c_{i+1}]$. To replace the condition on the variables $\mathbf{v}^i$, we will re-index these variables by $\mathbf{u}^1, \dots, \mathbf{u}^\ell$, where $\mathbf{u}^i = \sum_{j=1}^{i} \mathbf{v}^j$ for $i \in [\ell - 1]$ and $\mathbf{u}^\ell = \mathbf{p} - \mathbf{u}^{\ell-1}$, $\mathbf{u}^0 = \mathbf{0}$. Therefore, we have

$$T_\ell = \sum_{\substack{c_1,\dots,c_\ell \in \{0,\dots,k\} \\ \mathbf{u}^1,\dots,\mathbf{u}^{\ell-1} \in U}} |\mathrm{sub}(P[\mathbf{p} - \mathbf{u}^{\ell-1}], \mathcal{B}[c_\ell, c_1])| \prod_{i=1}^{\ell-1} [c_i < c_{i+1}] \cdot [\mathbf{u}^{i-1} \leq \mathbf{u}^i]$$

$$\cdot |\mathrm{sub}(P[\mathbf{u}^i - \mathbf{u}^{i-1}], \mathcal{B}[c_i, c_{i+1}])|.$$

By Lemma 4, we can compute T_ℓ in time $\ell \cdot (k \cdot \sigma_0(P))^2$ if we are given $|\mathrm{sub}(P[\mathbf{u}], \mathcal{B}[a, b])|$ for all $\mathbf{u} \in U$, $a, b \in \{0, \dots, k\}$.

It remains to show how we can compute $|\mathrm{sub}(P[\mathbf{u}], \mathcal{B}[a, b])|$ for given $\mathbf{u} \in U$, $a, b \in \{0, \dots, k\}$. Let $C_1, \dots, C_d$ be the connected components of $\mathcal{B}[a, b]$. Note that each C_i can be drawn in a $(k \times k)$-box with ply p, so we can use the oracle to compute $|\mathrm{sub}(P[\mathbf{w}], C_i)|$ for all $\mathbf{w} \in U$, $i \in [d]$. We would like to distribute the connected components of $P[\mathbf{u}]$ to $C_1, \dots C_d$. We can do this by dynamic programming. For $i \in [d]$ and $\mathbf{w} \in U$, we define

$$T'[i, \mathbf{w}] = |\mathrm{sub}(P[\mathbf{w}], C_1 \cup \dots \cup C_i)|$$

The recurrence is as follows:

$$T'[i, \mathbf{w}] = \sum_{\mathbf{w}' \leq \mathbf{w}} |\mathrm{sub}(P[\mathbf{w}'], C_i)| \cdot T[i - 1, \mathbf{w} - \mathbf{w}'],$$

where $\mathbf{w} \leq \mathbf{w'}$ indicates that $\mathbf{w}$ is in each coordinate smaller than $\mathbf{w'}$. Thus we can compute $T'[i, \mathbf{u}]$ in time $d|U|^2 = d\sigma_0(P)^2 \leq (n/k)\sigma_0(P)^2$.

Therefore, we can compute $|\mathrm{sub}(P, G)|$ in time $n \cdot \mathrm{poly}(k) \cdot \sigma_0(P)^2$ for host graphs that can be drawn in a box of width k with ply p. □

Theorem 4. *For unit disk graphs P and G with given embedding of ply p, $|\mathrm{sub}(P, G)|$ can be computed in time $\sigma_0(P)^2 \cdot n \cdot \mathrm{poly}(k)$ when given access to an oracle that computes $|\mathrm{sub}(P, G)|$ where the host graph has size $O(k^2 p)$ in constant time. In particular, there is a Turing kernel for computing $|\mathrm{sub}(P, G)|$ when $\sigma_0(P)$ and p are polynomial in k.*

Proof. By Lemma 6, if G can be drawn in a box of width k with ply p, we can compute $|\mathrm{sub}(P, G)|$ in time $n \cdot \mathrm{poly}(k) \cdot \sigma_0(P)^2$. If not, we use the same approach as in the proof of Lemma 6. The obtained building blocks will be disjoint unions of subgraphs that can be drawn in a box of width k with ply p. Using dynamic programming and applying Lemma 6, we conclude that we can compute $|\mathrm{sub}(P, G)|$ in time $n \cdot \mathrm{poly}(k) \cdot \sigma_0(P)^2$. □

4 Proof of Theorem 1: The Algorithm

We present only the proof for $\text{sub}(P, G)$, since the proof for $\text{ind}(P, G)$ is analogous. Before we start with the proof, we need to give a number of definitions: Suppose that a unit disk embedding of G in a $(b \times h)$-box with ply p is given. For integers $0 \le x \le x' \le b$, we define $G\langle x, x'\rangle$ as the induced subgraph of G whose vertex set consists are all vertices of G associated with disks in the unit disk embedding that are (partially) in between vertical lines x and x', i.e. the set of all disks that intersect the set $\{(a, b) \in \mathbb{R}^2 : x \le a \le x'\}$. We denote $G\langle x\rangle = G\langle x, x\rangle$.

Given functions $f_1 : D_1 \to R_1$ and $f_2 : D_2 \to R_2$, we say f_1 and f_2 are *compatible* if

- for all $u \in D_1 \cap D_2$, $f_1(u) = f_2(u)$, and
- for all $r \in R_1 \cap R_2$, we have $f_1^{-1}(r) = f_2^{-1}(r)$.

If f_1, f_2 are compatible, we define $f = f_1 \cup f_2$ as $f|_{D_1} = f_1$ and $f|_{D_2} = f_2$.

Note that in the above definition, the ranges of f_1 and f_2 matter. For example, the identity functions $f_1 : \{1\} \to \{1\}$ and $f_2 : \{2\} \to \{2\}$ are compatible, but the same functions are not compatible if we replace both ranges with $\{1, 2\}$.

Using Theorem 4, we can assume that G can be drawn in a $O(k) \times O(k)$ box with ply p. We will use dynamic programming. We will first define the sets of partial solutions that are counted in this dynamic programming algorithm. For variables

- integers $0 \le x < x' \le b$,
- separation (A, B) of P of order at most $2\sqrt{pk}$,
- injective $f : A \cap B \to G\langle x\rangle \cup G\langle x'\rangle$ such that $|f^{-1}(G\langle x\rangle)|, |f^{-1}(G\langle x'\rangle)| \le \sqrt{pk}$

we define

$$T[x, x', (A, B), f] = \{g \in \text{sub}(P[A], G\langle x, x'\rangle) : g \text{ extends } f\}.$$

Note that T is indexed by *any* separation of P of order $2\sqrt{pk}$. We will later replace this with a set of non-isomorphic separations to obtain the claimed dependence $\sigma_{2\sqrt{pk}}(P)$ in the running time.

Informally, $T[x, x', (A, B), f]$ is the set of all patterns $P[A]$ in $G\langle x, x'\rangle$ such that f describes their behaviour on the "boundary" $G\langle x\rangle \cup G\langle x'\rangle$. We will now show how to compute the table entries. We consider two cases, depending on whether $x' - x$ is less than $\sqrt{k/p}$ or not.

Case 1: $x' - x \le \sqrt{k/p}$
Note that in this case, the pathwidth of $G\langle x, x'\rangle$ is $O(\sqrt{pk})$ by Lemma 1. Using Lemma 3, we can compute $|\text{sub}(P, G)|$ in time $(pk)^{\sqrt{pk}}\sigma_{O(\sqrt{pk})}(P)n^{O(1)}$.

Case 2: $x' - x > \sqrt{k/p}$
Let $g \in T[x, x', (A, B), f]$, and let Q be the image of g. For $m \in \{x+1, \ldots, x'-1\}$, we say that Q is *sparse at m* if $|Q \cap G\langle m\rangle| \le \sqrt{pk}$, i.e. the vertical line at m

intersects at most $\sqrt{pk}$ disks in Q. Since $|Q| \leq k$ and $x' - x > \sqrt{k/p}$, there is at least one m such that Q is sparse at m by the averaging principle. Therefore,

$$T[x, x', (A, B), f] = \bigcup_{m=x+1}^{x'-1} \{g \in T[x, x', (A, B), f] : \ g(A) \text{ is a sparse at } m\}.$$

By the inclusion-exclusion principle, $|T[x, x', (A, B), f]|$ is equal to

$$\sum_{\emptyset \subset X \subseteq \{x+1,\ldots,x'-1\}} (-1)^{|X|} \left|\{g \in T[x, x', (A, B), f] : \ g(A) \text{ is sparse at all } m \in X\}\right|.$$

Denoting $X = \{x_1, \ldots, x_\ell\}$, where $x_1 < \cdots < x_\ell$, we further rewrite this into

$$\sum_{\ell=1}^{x'-x-2} (-1)^\ell \sum_{x<x_1<\cdots<x_\ell<x'} |\{g \in T[x, x', (A, B), f] : g(A) \text{ is sparse at } x_1, \ldots, x_\ell\}|.$$

Now we claim that, since $Q \cap G\langle m \rangle$ is a separator of $G[Q]$, $|T[x, x', (A, B), f]|$ can be further rewritten to express it recursively as follows:

Claim.

$$|T[x, x', (A, B), f]| = \sum_{\ell=1}^{x'-x-2} (-1)^\ell \sum_{(*)} \prod_{i=0}^{\ell} |T[x_i, x_{i+1}, (A_i, B_i), f_i]|,$$

where we let $x_0 = x$ and $x_{\ell+1} = x'$ for convenience and the sum $(*)$ goes over

- integers $x < x_1 < \cdots < x_\ell < x'$,
- separations (A_i, B_i) of P of order $2\sqrt{pk}$ for each $i = 0, \ldots, \ell$, such that
  - $\cup_{i=0}^{\ell} A_i = A$, and
  - $A_i \setminus B_i$ and $A_j \setminus B_j$ are disjoint for each $0 \leq i < j \leq \ell$,
- functions $f_i : A_i \cap B_i \to G\langle x_i \rangle \cup G\langle x_{i+1} \rangle$ for each $i = 0, \ldots, \ell$ such that $f, f_1, \ldots, f_\ell$ are pairwise compatible and $|f_i^{-1}(G\langle x_i \rangle)|, |f_i^{-1}(G\langle x_{i+1} \rangle)| \leq \sqrt{pk}$.

Proof of Claim. To prove this claim, consider first a function $g \in T[x, x', (A, B), f]$ such that $g(A)$ is sparse at $x_1, \ldots, x_\ell$. We describe how to find the separations (A_i, B_i) and functions f_i that correspond to g (for an example, see Fig. 1). Let $A_i = g^{-1}(G\langle x_i, x_{i+1} \rangle)$, $B_i = (P - A_i) \cup g^{-1}(G\langle x_i \rangle) \cup g^{-1}(G\langle x_{i+1} \rangle)$. Note that, since $g(A)$ is sparse at x_i and x_{i+1}, (A_i, B_i) is a separation of order at most $2\sqrt{pk}$. It is easy to see that $\cup A_i = g^{-1}(G\langle x, x' \rangle) = A$. Also, note that $A_i \setminus B_i = g^{-1}(G\langle x_i, x_{i+1} \rangle) \setminus (g^{-1}(G\langle x_i \rangle) \cup g^{-1}(G\langle x_{i+1} \rangle))$, so for any $i \neq j$, $A_i \setminus B_i$ and $A_j \setminus B_j$ are disjoint. We define $f_i : A_i \cap B_i \to G\langle x_i \rangle \cup G\langle x_{i+1} \rangle$ as $f_i = g|_{A_i \cap B_i}$. By construction, $f, f_1, \ldots, f_\ell$ are pairwise compatible.

Conversely, given pairwise compatible functions $g_0, \ldots, g_\ell$ such that $g_i \in T[x_i, x_{i+1}, (A_i, B_i), f_i]$, we show how to construct a function $g \in T[x, x', (A, B), f]$. Since g_i are compatible, we can define $g = g_0 \cup \cdots \cup g_\ell : A \to G\langle x, x' \rangle$. Since f, g_1, g_ℓ are pairwise compatible, g extends f. It is easy to see that this correspondence is one to one, which proves the claim.

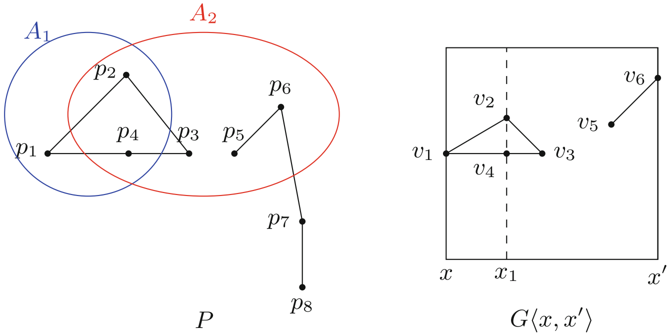

Fig. 1. The function $g : \{p_1, \ldots, p_6\} \to G\langle x, x'\rangle$ defined by $g(p_i) = v_i$, corresponds to functions $g_1 : A_1 \to G\langle x, x_1\rangle$ and $g_2 : A_2 \to G\langle x_1, x'\rangle$, where $g_1(p_i) = v_i$ and $g_2(p_i) = v_i$.

The next step is to rewrite the sum (∗) to match the form of Lemma 4. The only difference is that in (1) the summation is over variables that are pairwise independent.

Formally, let us define a square matrix M whose indices M_{ind} are of the form $(x_i, (A_i, B_i), f_i)$, where $x_i \in \{x, \ldots, x'\}$, $(A_i, B_i) \in \mathcal{S}_{2\sqrt{pk}}(P)$ and $f_i : A_i \cap B_i \to G\langle x_i\rangle \cup G\langle x_0\rangle$. Let $I_i = f_i^{-1}(G\langle x_i\rangle)$, $I_j = f_j^{-1}(G\langle x_j\rangle)$.

If $x_j \geq x_i$, f_i, f_j compatible and $|I_i|, |I_j| \leq \sqrt{pk}$ we define $M[(x_i, (A_i, B_i), f_i), (x_j, (A_j, B_j), f_j)]$ as

$$\mu((A_i, B_i), (A_j, B_j)) \cdot T[x_i, x_j, ((A_j \setminus A_i) \cup I_i, B_j \cup A_i), f_i|_{I_i} \cup f_j|_{I_j}],$$

and zero otherwise.

Intuitively, $M[(x_i, (A_i, B_i), f_i), (x_j, (A_j, B_j), f_j)]$ describes the number of ways to embed $P[A_j \setminus A_i]$ between lines x_i and x_j, where f_i and f_j describe the behaviour of these embeddings on lines x_i an x_j respectively. We observe that we can group isomorphic separations together, i.e. that instead of indexing by every separation, we can index by their representatives and take into account the multiplicities, which are described by μ.

Now we can rewrite the sum (∗) as

$$\sum_{(**)} M[(x_\ell, (A \setminus A_{\ell-1}, B \setminus B_{\ell-1}), f_{\ell-1}), (x', (A, B), f)]$$
$$\prod_{i=0}^{\ell-2} M[(x_i, (A_i, B_i), f_i), (x_{i+1}, (A_{i+1}, B_{i+1}, f_{i+1}))],$$

where the sum (∗∗) goes over $(x_0, (A_0, B_0), f_0), \ldots, (x_{\ell-1}, (A_{\ell-1}, B_{\ell-1}), f_{\ell-1}) \in M_{ind}$. Now by Lemma 4, we can compute the sum (∗) in time $\ell \cdot |M_{ind}|^2$. Let us bound the size of M_{ind}. Recall that $|\mathcal{S}_{2\sqrt{pk}}(P)| = \sigma_{2\sqrt{pk}}(P)$ and note that $G\langle x_i\rangle$ contains at most $k^2 p$ disks (since we can assume that G can be drawn in a $(O(k^2) \times O(k^2))$-box with ply p by Theorem 4). Thus we have

$$|M_{ind}| \leq k^2 \sigma_{2\sqrt{pk}}(P) \cdot (k^2 p)^{\sqrt{pk}},$$

Therefore, we can compute $|\text{sub}(P, G)|$ in time $k^{O(\sqrt{pk})} \cdot p^{O(\sqrt{pk})} \cdot \sigma_{O(\sqrt{pk})}(P)^2$.

5 Theorem 3: Lower Bound

In this section, we give a proof overview of Theorem 3, showing that under ETH there is no algorithm deciding whether $|\text{sub}(P, G)| > 0$ ($|\text{ind}(P, G)| > 0$ respectively) in time $2^{o(n/\log n)}$ even when the ply is two. The formal proof can be found in the full version. We will use a reduction from the String 3-Groups problem similar to the one in [4].

Definition 3. *The* String 3-Groups *problem is defined as follows. Given sets* $A, B, C \subseteq \{0,1\}^{6\lceil \log n \rceil + 1}$ *of size* n*, find* n *triples* $(a, b, c) \in A \times B \times C$ *such that for all* i*,* $a_i + b_i + c_i \leq 1$ *and each element of* A, B, C *occurs exactly once in a chosen triple.*

We call the elements of A, B, C strings. It was shown in [4] that, assuming the ETH, there is no algorithm that solves String 3-Groups in time $2^{o(n)}$. Given an instance (A, B, C) of String 3-Groups problem, we construct the corresponding host graph G and pattern P. Firstly, we modify slightly the strings in A, B, C to facilitate the construction of P and G. Let m be the length of the (modified) strings. For each $a \in A$, the connected component in G that corresponds to it consists of two paths $p_1 \dots p_m$ and $q_1 \dots q_m$, where p_i and q_i are connected by paths of length 3 if $a_i = 0$. For each $b \in B$, the connected component in P that corresponds to it consists of a path $t_1 \dots t_m$, where there is a path of length two attached to t_i if $b_i = 1$. The connected components corresponding to elements in C are constructed in a similar way. Finally, we add gadgets (triangles and 4-cycles) to each connected component in P and G to ensure we cannot "flip" the components in P. For an example, see Fig. 2.

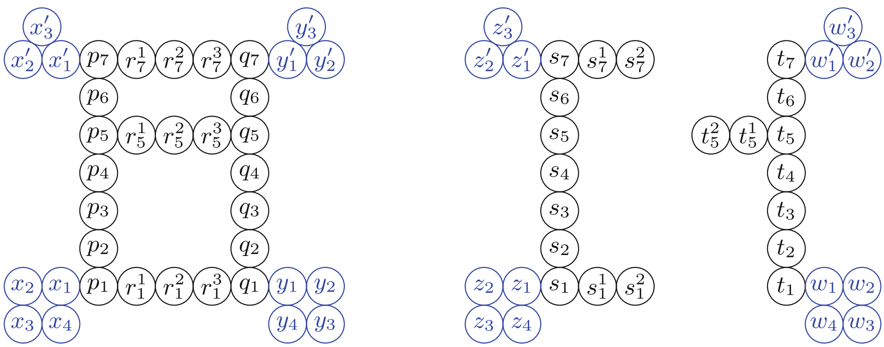

Fig. 2. Connected components corresponding to $a = 0111010 \in A$ (left), $b = 1000001 \in B$ (middle), $c = 0000100 \in C$ (right).

6 Concluding Remarks

We gave (mostly) sub-exponential parameterized time algorithms for computing $|\text{sub}(P, G)|$ and $|\text{ind}(P, G)|$ for unit disk graphs P and G. Since the fine-grained parameterized complexity of the subgraph isomorphism problem was only recently understood for planar graphs, we believe our continuation of the study for unit disk graphs is very natural, we hope it inspires further general results.

While our algorithms are tight in many regimes, they are not tight in all regimes. In particular, the (sub)-exponential dependence of the runtime in the ply is not always necessary: We believe the answer to this question may be quite complicated: For detecting some patterns, such as paths, $2^{O(\sqrt{k})}n^{O(1)}$ time algorithms are known [16], but it seems hard to extend it to the counting problem (and to all patterns with few non-isomorphic separations of small order).

For counting induced occurrences with bounded clique size our method can be easily adjusted to get a better dependence in the ply: I.e. our method can be used to a get a $(kp)^{O(\sqrt{k})}$ time algorithm for counting independent sets of size k in unit disk graphs of ply p (which is optimal under the ETH by [11]); is there such an improved independence on the ply for each pattern P?

Finally, we note it would be interesting to study the complexity of computing $|\text{sub}(P, G)|$ and $|\text{ind}(P, G)|$ for various pattern classes and various other geometric intersection graphs as well. Our results can be adapted to disk graphs where the ratio of the largest and smallest radius is constant (using a slight modification of Lemma 2). A possible direction for further research would be to determine for which patterns can one compute the above values on bounded ply disk graphs? Recent work [10] shows some problems admit algorithms running in sub-exponential time parameterized time.

References

1. Baker, B.S.: Approximation algorithms for np-complete problems on planar graphs. J. ACM (JACM) **41**(1), 153–180 (1994)
2. de Berg, M., Bodlaender, H.L., Kisfaludi-Bak, S., Marx, D., van der Zanden, T.C.: A framework for exponential-time-hypothesis-tight algorithms and lower bounds in geometric intersection graphs. SIAM J. Comput. **49**(6), 1291–1331 (2020). https://doi.org/10.1137/20M1320870
3. Bhore, S., Carmi, P., Kolay, S., Zehavi, M.: Parameterized study of steiner tree on unit disk graphs. Algorithmica **85**(1), 133–152 (2023). https://doi.org/10.1007/s00453-022-01020-z
4. Bodlaender, H.L., Nederlof, J., Zanden, T.C.V.D.: Subexponential time algorithms for embedding H-minor free graphs, vol. 55. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, August 2016. https://doi.org/10.4230/LIPIcs.ICALP.2016.9
5. Borgs, C., Chayes, J., Kahn, J., Lovász, L.: Left and right convergence of graphs with bounded degree. Random Struct. Algorithms **42**, 1–28 (2013). https://doi.org/10.1002/rsa.20414

6. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
7. Fomin, F.V., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering. SIAM J. Comput. **51**(6), 1866–1930 (2022). https://doi.org/10.1137/19m1262504
8. Har-Peled, S., et al.: Stabbing pairwise intersecting disks by five points. Discret. Math. **344**(7), 112403 (2021)
9. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. J. ACM (JACM) **32**(1), 130–136 (1985)
10. Lokshtanov, D., Panolan, F., Saurabh, S., Xue, J., Zehavi, M.: Subexponential parameterized algorithms on disk graphs (extended abstract). In: Naor, J.S., Buchbinder, N. (eds.) Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference/Alexandria, VA, USA, 9–12 January 2022, pp. 2005–2031. SIAM (2022). https://doi.org/10.1137/1.9781611977073.80
11. Marx, D.: On the optimality of planar and geometric approximation schemes. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), 20–23 October 2007, Providence, RI, USA, Proceedings, pp. 338–348. IEEE Computer Society (2007). https://doi.org/10.1109/FOCS.2007.50
12. Marx, D., Pilipczuk, M.: Subexponential parameterized algorithms for graphs of polynomial growth. In: Pruhs, K., Sohler, C. (eds.) 25th Annual European Symposium on Algorithms, ESA 2017, 4–6 September 2017, Vienna, Austria. LIPIcs, vol. 87, pp. 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ESA.2017.59
13. Nederlof, J.: Detecting and counting small patterns in planar graphs in subexponential parameterized time. In: Makarychev, K., Makarychev, Y., Tulsiani, M., Kamath, G., Chuzhoy, J. (eds.) Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, 22–26 June 2020, pp. 1293–1306. ACM (2020). https://doi.org/10.1145/3357713.3384261
14. Nešetřil, J., De Mendez, P.O.: Sparsity: Graphs, Structures, and Algorithms, vol. 28. Springer Science & Business Media, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27875-4
15. Panolan, F., Saurabh, S., Zehavi, M.: Contraction decomposition in unit disk graphs and algorithmic applications in parameterized complexity. In: Chan, T.M. (ed.) Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, 6–9 January 2019, pp. 1035–1054. SIAM (2019). https://doi.org/10.1137/1.9781611975482.64
16. Zehavi, M., Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: ETH-tight algorithms for long path and cycle on unit disk graphs. J. Comput. Geom. **12**(2), 126–148 (2021). https://doi.org/10.20382/jocg.v12i2a6