Algorithmica manuscript No.
(will be inserted by the editor)

Optimality program in segment and string graphs

Édouard Bonnet · Paweł Rzążewski

Received: date / Accepted: date

Abstract Planar graphs are known to allow subexponential algorithms running in time $2^{O(\sqrt{n})}$ or $2^{O(\sqrt{n}\log n)}$ for most of the paradigmatic problems, while the brute-force time $2^{\Theta(n)}$ is very likely to be asymptotically best on general graphs. Intrigued by an algorithm packing curves in $2^{O(n^{2/3}\log n)}$ by Fox and Pach [SODA'11], we investigate which problems have subexponential algorithms on the intersection graphs of curves (string graphs) or segments (segment intersection graphs) and which problems have no such algorithms under the ETH (Exponential Time Hypothesis). Among our results, we show that, quite surprisingly, 3-Coloring can also be solved in time $2^{O(n^{2/3}\log^{O(1)} n)}$ on string graphs while an algorithm running in time $2^{o(n)}$ for 4-Coloring even on axis-parallel segments (of unbounded length) would disprove the ETH. For 4-Coloring of unit segments, we show a weaker lower bound, excluding a $2^{o(n^{2/3})}$ algorithm (under the ETH). The construction exploits the celebrated Erdős-Szekeres theorem. The subexponential running time also carries over to Min Feedback Vertex Set, but not to Min Dominating Set and Min Independent Dominating Set.

1 Introduction

Most combinatorial optimization and decision problems admit subexponential algorithms when restricted to planar graphs. More precisely, they can be solved in time $2^{O(\sqrt{n})}$, or $2^{\tilde{O}(\sqrt{n})}$ on planar graphs with n vertices, while under

The extended abstract of this paper was presented at WG 2018 [8].

É. Bonnet
ENS Lyon, LIP, Lyon, France, E-mail: edouard.bonnet@dauphine.fr

P. Rzążewski
Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland, E-mail: p.rzazewski@mini.pw.edu.pl

the ETH (Exponential Time Hypothesis, which asserts that 3-Sat cannot be solved in subexponential time [26, 27]) they do not admit an algorithm running in time $2^{o(n)}$ on general graphs. The former is due to the facts that planar graphs have treewidth $O(\sqrt{n})$ and that we have efficient algorithms parameterized by the treewidth tw of the graph, namely running in $2^{O(\text{tw})}n^{O(1)}$, or $2^{\tilde{O}(\text{tw})}n^{O(1)}$.

The so-called bidimensionality theory [13, 15–17] pushes this square-root phenomenon further by yielding $2^{O(\sqrt{k})}n^{O(1)}$ algorithms where k is the targeted size of a solution (think for example of the problems of finding a maximum independent set or a minimum dominating set of size k). In a nutshell, it exploits a deep structural result by Robertson, Seymour, and Thomas [40]: planar graphs with treewidth tw have a $\Theta(\text{tw})$-by-$\Theta(\text{tw})$ grid as a minor (i.e., any graph obtained by deleting vertices and edges, and contracting edges). Thus, if the presence of a large grid minor makes the problem trivial (as in, one can always answer yes or always answer no), then one only has to solve efficiently instances with low treewidth; which, as we noted, can often be done. The claimed running time is obtained by defining large grids as $\Theta(\sqrt{k})$-by-$\Theta(\sqrt{k})$, since their absence as minors implies that the treewidth is in $O(\sqrt{k})$. The bidimensionality theory is also used to obtain approximation schemes and linear kernels and could be generalized to graphs with bounded genus and graphs excluding a fixed minor [14].

A natural line of research is to generalize or extend the subexponential (parameterized) algorithms to classes of graphs which do not fall into those categories. For geometric intersection graphs, the situation is much richer than for planar graphs. For instance, Marx and Pilipczuk already observed that packing problems (of the kind of Max Independent Set) are more broadly subject to subexponential algorithms – running typically in $n^{O(\sqrt{k})}$ – than covering problems (of the kind of Min Dominating Set) – for which $n^{O(k)}$ is often essentially optimal under the ETH [35, 36].

We briefly survey the existing results in the design of subexponential algorithms on geometric intersection graphs. A prominent role is played by intersection graph of families of *fat objects*, i.e., objects for which the aspect ratio (their length divided by their width) is bounded. We highlight that fat objects, and in particular disks and squares, often allow faster algorithms and the so-called square-root phenomenon. As we will see, subexponential algorithms are less frequent on intersection graphs of curves and segments but nevertheless present such as exemplified by Max Independent Set, 3-Coloring, Min Feedback Vertex Set, and Maximum Induced Matching.

Subexponential algorithms on geometric intersection graphs. By a *ply* of a family of geometric objects we denote the maximum number of objects covering a single point. Smith and Wormald show that for any collection of n convex fat objects with ply p there is a balanced separator of size $O(\sqrt{np})$ [44]. This leads to subexponential algorithms when the ply is constant, or in general for problems becoming trivial when the ply is too large, such as k-Coloring. The

$2^{\tilde{O}(\sqrt{nk})}$-time algorithm that this win-win provides for coloring n fat objects, say disks, with k colors is shown essentially optimal under the ETH by Biró et al. [5].

A next step may consist of designing FPT[1] or XP[2] algorithms where the dependency in the parameter is subexponential (for problems of the form "find k vertices such that..."). Using a shifting argument à la Baker [4], Alber and Fiala obtain a $n^{O(\sqrt{k})}$-time algorithm to decide if one can find k disjoint unit disks or squares among n [3]. Marx and Pilipczuk generalize this result to packing k disjoint polygons among n in the same time [35, 36]. Their approach is based on guessing a small separator in the medial axis (i.e., the Voronoi diagram of polygons) of a supposed solution, as suggested by Adamaszek and Wiese and Har-Peled to obtain QPTAS for geometric packing problems [1, 2, 25].

Marx showed that Max Independent Set and Min Dominating Set in the intersection graphs of disks or squares are W[1]-hard, and therefore unlikely to be FPT [34]. Those reductions also show that the $n^{O(\sqrt{k})}$ algorithms [35, 36] are essentially optimal under the ETH. Fomin et al. [20] observed that unit disks of bounded degree have treewidth $O(\sqrt{n})$ and used this fact to extend bidimensionality to unit disk graphs for a handful of problems. Recently, a superset of the previous authors gave $2^{O(\sqrt{k})}n^{O(1)}$-time algorithms for k-Feedback Vertex Set, k-Path, k-Cycle, Exact k-Cycle [19].

Non-fat objects: segments and strings. Segment intersection graphs (or segment graphs in short) are the intersection graphs of straight-line segments in the plane. They are called unit segments if all the segments of a representation share the same length. For a fixed integer k, k-Dir is defined as the set of intersection graphs of segments, each parallel to one of fixed k directions. Strings graphs are the intersection graphs of simple curves in the plane. Those curves can be assumed polygonal without loss of generality. The vertices of the polygonal curves in a geometric representation are called *geometric vertices* not to confuse them with the actual vertices of the graph. As shown by Kratochvíl and Matoušek, there are string graphs with n vertices, which require $2^{\Omega(n)}$ geometric vertices in any string representation with polygonal curves [30].

A systematic study of segment graphs and their subclasses was initiated by Kratochvíl and Matoušek [31]. It is interesting to point out that every planar graph is a segment graph, as shown by Chalopin and Gonçalves [12] (this was a long-standing conjecture by Scheinerman [43], see also a recent proof by Gonçalvez, Isenmann, and Pennarun [24]).

The class of string graphs is very general, as it includes split graphs (i.e., graphs whose vertices can be partitioned into two sets inducing a clique and an independent set), intersection graphs of bodies (i.e., compact shapes with non-empty interior), or incomparability graphs (i.e., graphs whose vertex set

[1] with running time $f(k)n^{O(1)}$

[2] with running time $n^{f(k)}$

is given by the set of elements of a poset, and edges join elements that are incomparable).

Biró et al. showed that even though coloring disks or more generally fat objects with a constant number of colors can be solved in $2^{\tilde{O}(\sqrt{n})}$ [5], 6-coloring axis-parallel segments (2-Dir) in time $2^{o(n)}$ would refute the ETH. This suggests that subexponential algorithms are less frequent on the intersection graphs of non-fat objects such as segments and strings. On the other hand, Fox and Pach presented a subexponential algorithm for Max Independent Set on string graphs [21]. Their approach uses a win-win strategy and is based on the existence of balanced separators in string graphs. Fox, Pach, and Tóth showed that string graphs with m edges have balanced separators of size $O(m^{3/4} \log m)$, and conjectured that there is always a separator of size $O(\sqrt{m})$ [23]. Matoušek showed that string graphs admit a balanced separator of size $O(\sqrt{m} \log m)$ [38]. Finally, recently Lee improved the result of Matoušek, proving the conjecture.

Theorem 1 (Lee [32]) *Every string graph with m edges has a balanced separator of size $O(\sqrt{m})$. Moreover, it can be found in polynomial time, provided that the geometric representation is given.*

Let us point out that this result generalizes the famous planar separator theorem by Lipton and Tarjan [33], as planar graphs are string graphs and the number of edges in a planar graph is linear in the number of vertices. This also shows that Theorem 1 is best possible (up to the constants), as the planar separator theorem is asymptotically tight.

Our contributions. We show that the subexponential algorithm for Max Independent Set in string graphs by Fox and Pach [21], running in time $2^{\tilde{O}(n^{2/3})}$, can be extended to 3-Coloring and Min Feedback Vertex Set. As in the algorithm of Fox and Pach, the central idea is a win-win: either the graph is rather sparse and the separator of Theorem 1 gives a speed-up, or the graph has a high-degree vertex (used for 3-Coloring) or a large biclique (used for Min Feedback Vertex Set) and an efficient branching can be performed. Refining a lower bound of Biró et al. [5], we complement this former result by showing that for any $k \geqslant 4$, k-Coloring cannot be solved in $2^{o(n)}$ even on axis-parallel segments, unless the ETH fails. The reduction relies on having segment lengths with two different orders of magnitude. We therefore ask if unit segments could allow a faster algorithm for k-Coloring for $k \geqslant 4$. Under the ETH, we provide a stronger lower bound than the one for planar graphs (which refutes a running time $2^{o(\sqrt{n})}$) and show that unit segments cannot be k-colored in $2^{o(n^{2/3})}$ for any $k \geqslant 4$. Our construction uses the fact, closely related to the famous Erdős-Szekeres [18] theorem, that any permutation on n totally ordered elements can be partitioned into $O(\sqrt{n})$ monotone subsequences (see Knuth [28, Sec. 5.1.4]).

We then give tight ETH lower bounds for Min (Connected) Dominating Set and Min Independent Dominating Set on segment graphs and

Max Clique on string graphs. For that, we design reductions whose number n of produced segments is linear in $N + M$ from satisfiability problems with N variables and M clauses. Indeed, the sparsification lemma of Impagliazzo et al. [27] implies that those satisfiability problems are not solvable in $2^{o(N+M)}$ unless the ETH fails; which enables us to conclude that the problems are not solvable in $2^{o(n)}$ under the ETH, on graphs with n vertices.

Although the NP-hardness of the mentioned problems is known for segment intersection graphs [11, 46], getting such linear reductions might be difficult.

For instance, while it is known that planar graphs are a subclass of segment intersection graphs [12], implying the NP-hardness of all the problems of Table 1 except k-Coloring for $k \geqslant 4$ and Max Clique, this fact does not serve our purpose since they can be solved in time $2^{O(\sqrt{n})}$ on planar graphs. The situation is an interesting intermediate between planar and general graphs. Our objects *can* intersect but we cannot afford crossover gadgets (at least not quadratically many). Certain intersections create unwanted edges, whose importance we have to tame. It is also noteworthy that segment/string graphs cannot be expanders since if they have constant degree, by Theorem 1, they have treewidth $\tilde{O}(\sqrt{n})$. Hence, we are deprived of the *usual hardest instances.*

Problem	Upper bound	Lower bound
Max Independent Set	$2^{\tilde{O}(\sqrt{n})}p^{O(1)}$, $2^{\tilde{O}(n^{2/3})}$	$2^{\Omega(\sqrt{n})}$
3-Coloring	$2^{\tilde{O}(n^{2/3})}$	$2^{\Omega(\sqrt{n})}$
k-Coloring for every $k \geqslant 4$	$2^{O(n)}$	$2^{\Omega(n)}$ (even in 2-Dir)
k-Coloring for every $k \geqslant 4$	$2^{O(n)}$	$2^{\Omega(n^{2/3})}$ in unit 3-Dir
Min Feedback Vertex Set	$2^{\tilde{O}(n^{2/3})}$	$2^{\Omega(\sqrt{n})}$
Maximum Induced Matching	$2^{\tilde{O}(n^{2/3})}$	$2^{\Omega(\sqrt{n})}$
Min (Connected) Dominating Set	$2^{O(n)}$	$2^{\Omega(n)}$
Min Independent Dominating Set	$2^{O(n)}$	$2^{\Omega(n)}$
Max Clique	$2^{O(n)}$	$2^{\Omega(n)}$ (on string graphs)

Table 1 Upper and lower bounds for classical problems on string and segment graphs. The **upper bounds** work on **string graphs**. The **lower bounds** are conditional on the ETH, and designed on **segment graphs**, unless stated otherwise. New results are indicated by the shaded background. By p we denote the number of geometric vertices if a geometric representation is given.

Geometric representation and robust algorithms. In case of graphs with geometric representations, it is important to distinguish between a graph itself (i.e., a pure abstract structure, for which we know that some geometric representation exists), and the representation itself. Note that this is not the case with planar graphs, as finding a plane embedding can be done in linear time [9].

Finding a segment or string representation of a graph was shown to be NP-hard by Kratochvíl [29], and Kratochvíl and Matoušek [31], respectively. However, it was very unclear if the problems are in NP (which is usually the trivial part of an NP-completeness proof). As mentioned above, Kratochvíl and

Matoušek [30] showed that some string graphs require a representation of exponential size, which proved that the simple idea of exhaustively guessing the representation cannot work for this problem. Finally, the NP-membership of recognizing string graphs was proven by Schaefer, Sedgwick, and Štefankovič [41].

The story of recognizing segment graphs is even more interesting. On the first sight, the situation seems simpler than for strings, as the number of geometric points in a segment representation is clearly polynomial in n. However, there are segment graphs, whose every segment representation requires points with coordinates doubly exponential in n, i.e., using $2^{\Omega(n)}$ digits (see Kratochvíl and Matoušek [31], and McDiarmid and Müller [39]). Finally, the problem was shown to be complete for the class $\exists\mathbb{R}$ (see Schaefer and Štefankovič [42]), i.e., the class of problems reducible in polynomial time to deciding if a given existential formula over the reals is true. This is a strong evidence that the problem is not in NP. For a nice exposition of the $\exists\mathbb{R}$-completeness proof, see Matoušek [37].

All this shows that a requirement of an explicit geometric representation of an input graph may be a serious drawback of an algorithm. We call an algorithm *robust* if it takes only an abstract structure like a graph as an input, and either computes the solution, or concludes (correctly) that the input graph does not belong to the desired class. On the one hand, our algorithms (see Section 2) are robust, but work slightly faster if the input is given along with the geometric representation. On the other hand, the lower bounds (see Section 3) hold even if the geometric representation is given explicitly.

2 Upper bounds

Fox and Pach showed that, on string graphs, a maximum independent set can be computed in subexponential time:

Theorem 2 (Fox & Pach [21], Lee [32]) Max Independent Set *can be solved in time* $2^{O(n^{2/3}\log n)}$ *in string graphs with* n *vertices.*

In their paper, they give a worse running time than the one claimed above. This is because they used the $O(m^{3/4}\log m)$ separator theorem [23], which has been recently improved to $O(\sqrt{m})$ [32]. The algorithm is a simple win-win argument. If there is a vertex with degree at least $n^{1/3}$, then either removing it or selecting it and removing its neighbors gives a branching $F(n) \leqslant F(n-1) + F(n - \lceil n^{1/3}\rceil - 1)$. Otherwise, if all the vertices have degree smaller than $n^{1/3}$, the graph is rather sparse and the balanced separator of size $O(\sqrt{m}) = O(n^{2/3})$ provides an efficient divide-and-conquer. The threshold $n^{1/3}$ is computed so that it balances the running time of those two subroutines and gives the claimed overall asymptotic time.

This result was somewhat improved by Marx and Pilipczuk [35, 36] based on an approach introduced by Adamaszek, Har-Peled, and Wiese [1]. However, their algorithm requires that the string graph is given with a representation by polygonal curves on a polynomial number of geometric vertices.

Theorem 3 (Marx & Pilipczuk [35]) MAX INDEPENDENT SET *can be solved in time* $2^{O(\sqrt{n}\log n)}p^{O(1)}$ *in string graphs with* n *vertices, where the strings are given as polygonal curves on a total of* p *geometric vertices.*

In a nutshell, the idea is to exhaustively guess a small balanced face-separator in the Voronoi diagram of a supposed (although not known) fixed solution, and solve recursively the two subinstances in the inside and outside of this separator.

Let us point out that since the complement of an independent set is a vertex cover, the algorithms from Theorems 2 and 3 can be used to solve the MIN VERTEX COVER problem within the same time bounds.

If the approach of Marx and Pilipczuk does not seem to generalize easily to coloring problems, the win-win of Fox and Pach can be transported to 3-COLORING with a bit more arguments. The algorithm even works for the more general LIST 3-COLORING (in LIST k-COLORING each vertex v is equipped with a list $L(v) \subseteq [k]$ and we want to find a proper coloring, in which every vertex receives a color from its list).

Theorem 4 LIST *3*-COLORING *of a string graph with* n *vertices can be decided in time* $2^{O(n^{2/3}\log n)}$*, even without geometric representation.*

Proof. Consider an instance (G, L) of LIST 3-COLORING with n vertices. Observe that without loss of generality we can assume that each list has two or three elements. Indeed, if there is a vertex with just one allowed color, we can fix this color and remove it from the list of each of its neighbors. Let N be the sum of the lengths of the lists; clearly $2n \leqslant N \leqslant 3n$.

First, assume that G has no vertex with degree larger than $n^{1/3}$, then the number m of edges is $O(n^{4/3})$. By Theorem 1, G has a balanced separator of size $O(\sqrt{m}) = O(n^{2/3})$. We can find this separator in polynomial time, if the representation is given, or by exhaustive guessing in time $n^{O(n^{2/3})} = 2^{O(n^{2/3}\log n)}$, without using a representation. Then we list all possible colorings of the separator and proceed with a standard divide-and-conquer approach. The total time complexity of this step is $2^{O(n^{2/3}\log n)}$.

If there is a vertex v of degree at least $n^{1/3}$, then one among the lists: $\{1,2\}, \{1,3\}, \{2,3\}$, $\{1,2,3\}$ appears on at least $n^{1/3}/4$ of its neighbors. Thus there are two colors (say, 1 and 2) that appear in lists of at least $n^{1/3}/4$ of neighbors of v. Since the list of v has size at least two, one of these colors (say 1) appears on the list of v. We branch into two possibilities: choosing the color 1 for v (then we exclude 1 from the lists of all the neighbors of v), and not choosing 1 for v (then we remove 1 from the list of v). The complexity F of this step is given by the recursion

$$\begin{aligned}F(N) \leqslant & F(N - n^{1/3}/4) + F(N-1) \leqslant F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right) + F(N-1)\\ \leqslant & F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right) + F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right) + F(N-2)\\ \leqslant & 3\cdot F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right) + F(N-3) \leqslant \dots\\ \leqslant & \left(N^{1/3}/(3^{1/3}\cdot 4)\right)\cdot F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right) + F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right)\\ = & \left(N^{1/3}/(3^{1/3}\cdot 4)+1\right)\cdot F\left(N - N^{1/3}/(3^{1/3}\cdot 4)\right).\end{aligned}$$

This inequality is satisfied by

$$F(N) = \left(N^{1/3}/(3^{1/3}\cdot 4)+1\right)^{\overline{\frac{N}{N^{1/3}/(3^{1/3}\cdot 4)}}} = 2^{O(N^{2/3}\log N)} = 2^{O(n^{2/3}\cdot\log n)}.$$

Combining these two cases gives the claimed time complexity. Finally, observe that if the input graph is not a string graph, then the exhaustive search for a separator might fail, and then we can report a wrong input instance. □

In Min Feedback Vertex Set problem we ask for the minimum set of vertices, whose removal destroys all cycles in a graph. For this problem, there is no obvious subexponential branching on a high-degree vertex. Instead, we use the following theorem by Lee.

Theorem 5 (Lee [32]) *There is a constant c such that for any $t \geqslant 1$, $K_{t,t}$-free string graphs on n vertices have fewer than $c \cdot t\log t \cdot n$ edges.*

It is worth mentioning that Fox and Pach [22, Theorem 5] obtained a slightly weaker result with $\log^{O(1)} t$ instead of $\log t$. Theorem 5 is the last tool we need to show the following.

Theorem 6 Min Feedback Vertex Set *on string graphs with n vertices can be solved in time $2^{\tilde{O}(n^{2/3})}$.*

Proof. The proof is similar to the proof of Theorem 4, but it involves a slight technical complication. We will solve a more general problem, where the input is a graph G, a set $\mathcal{C}_1$ of constraints of type $\texttt{disconnect}(u,v)$, and another set $\mathcal{C}_2$ of constraints of type $\texttt{stays}(v)$, where u, v are vertices of G. We ask for a minimum feedback vertex set X of G, such that for every constraint $\texttt{disconnect}(u,v)$, the vertices u and v are in different connected components of $G - X$, and for every constraint $\texttt{stays}(v)$ we have $v \notin X$. The algorithm is recursive, the constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ are checked at the leaves of the recursion tree. Clearly, if $\mathcal{C}_1 = \mathcal{C}_2 = \emptyset$, then we just ask for the minimum feedback vertex set.

If G has fewer than $c/3 \cdot n^{4/3}\log n$ edges (where c is a constant from Theorem 5), then by Theorem 1 there is a balanced separator S of size $O(\sqrt{m}) = \tilde{O}(n^{2/3}\log^{1/2} n)$, we can find it in time $n^{O(n^{2/3})} = 2^{O(n^{2/3}\log^{3/2} n)}$

by exhaustive search or in polynomial time, if the geometric representation is given. We partition the vertices of of $G - S$ into sets V_1, V_2, such that $V_1, V_2 \leqslant c' \cdot n$ (for a constant c') and S separates V_1 from V_2; such sets exist since S is a balanced separator. We will exhaustively guess the intersection I' of a fixed minimum solution with S, taking into consideration the current constraints $\mathcal{C}_2$ (this represents at most $2^{|S|} = 2^{\tilde{O}(n^{2/3})}$ possibilities), introduce the new constraints $\texttt{stays}(v)$ for every $v \in S \setminus I'$, and solve the problem independently in $G_1 := G[V_1 \cup S \setminus I]$ and $G_2 := G[V_2 \cup S \setminus I]$.

However, note that there might be cycles in G that are not fully contained in $V_1 \cup S$ or in $V_2 \cup S$ and the straightforward approach discussed above would not destroy them. Let us call such cycles *essential.* For each essential cycle C, and for $i = 1, 2$, we call *i-subpath of C* a subpath of C, whose endvertices are in S, and inner vertices are in V_i. To destroy C, we must disconnect the endvertices of some i-subpath of C. We ensure this by introducing appropriate separation constraints. For every partition $\Pi = (S_1, S_2, \ldots, S_k)$ of $S \setminus I$, we run the algorithm recursively in each graph G_i with additional constraints $\texttt{disconnect}(u, v)$ for every $u, v \in S$, such that u and v are in different parts of Π. The number of partitions of $S \setminus I$ is given by the Bell number of $|S \setminus I|$, which is upper-bounded by $|S|^{|S|} = 2^{|S| \log |S|} = 2^{\tilde{O}(n^{2/3})}$. This gives us a total of $2^{\tilde{O}(n^{2/3})}$ recursive calls at each level of the recursion tree. It is sufficient to only consider those connectivity patterns since being connected is a transitive relation: if u and v, and v and w stay connected in G_i, then u and w also stay connected. We combine solutions in G_1 and G_2 which agree on the subset $I \subseteq S$, and such that the essential cycles cannot survive. It is known and relatively easy to see that this happens exactly when the partitions $\Pi^1 = (S^1_1, S^1_2, \ldots, S^1_{k_1})$ for G_1 and $\Pi^2 = (S^2_1, S^2_2, \ldots, S^2_{k_2})$ for G_2 are such that for each pair (i, j), $|S^1_i \cup S^2_j| \leqslant 1$ and the bipartite intersection graph (with an edge between S^1_i and S^2_j iff they have non-empty intersection) is a forest (see e.g., [6, 7]). This way, for every connectivity pattern in $S \setminus I$, we will find the minimum feedback vertex set respecting this pattern. When we combine the solutions of both subproblems, we reject the ones that do not agree on S, and those with an essential cycle C for which the endvertices of every i-subpath are not disconnected in G_i. This step has a total running time $2^{\tilde{O}(n^{2/3})}$.

On the other hand, if G has at least $c/3 \cdot n^{4/3} \log n$ edges, by Theorem 5, there is a subgraph of G isomorphic to $K_{n^{1/3}, n^{1/3}}$. We can find it by exhaustive guessing in time $n^{2n^{1/3}} \cdot poly(n) = 2^{\tilde{O}(n^{1/3})}$. Observe that any feedback vertex set of G must contain all but one vertex of one bipartition class of the biclique. This observation gives us a branching algorithm: we choose the vertex v which is *not* necessarily included in the the solution in $2n^{1/3}$ ways, then we remove all other vertices from the bipartition class of v, and proceed recursively. Note that v might be still chosen to the solution in next steps. The complexity of this algorithm is given by the recursion

$$F(n) \leqslant 2^{\tilde{O}(n^{1/3} \log n)} + 2n^{1/3} F(n - n^{1/3} + 1). \qquad (1)$$

Note that the depth of recursion is $O(n^{2/3})$, so the inequality is solved by

$$F(n) = (2n^{1/3})^{O(n^{2/3})} = 2^{O(n^{2/3} \log n)}.$$

We also trim the branches which violate a constraint of $\mathcal{C}_2$. Branching does not introduce new constraints. In particular, the vertex which is not added to the solution (while all the other vertices of its bipartition class) might still be included in the solution at a later stage. Observe that the branching on a separator and the branching on a biclique produce instances of the same problem and can be done in an interleaved fashion.

Finally, if the exhaustive search for a separator or a biclique fails, then either we have reached a constant size, in which case the subproblem can be brute-forced, or we can correctly report that the input graph is not a string graph. □

It is interesting to note that a similar approach works also for the Maximum Induced Matching problem.

Theorem 7 Maximum Induced Matching *on string graphs with n vertices can be solved in time $2^{\tilde{O}(n^{2/3})}$.*

Proof. Again, to ensure consistency of solutions found in recursive calls, we will solve a more general problem, where we are given a graph G and a subset X of its vertices, and we ask for a maximum induced matching which covers all vertices from X. Clearly, if $X = \emptyset$, then we just ask for a maximum induced matching. Whenever in the recursion a neighbor u of a vertex $v \in X$ becomes matched to some vertex other than v, we can immediately terminate the current recursive call, as it will not produce a feasible solution.

If G has fewer than $c/3 \cdot n^{4/3} \log n$ edges (where c is a constant from Theorem 5), then by Theorem 1 there is a balanced separator S of size $O(\sqrt{m}) = \tilde{O}(n^{2/3} \log^{1/2} n)$, which can be found in time $2^{O(n^{2/3} \log^{3/2} n)}$ by exhaustive search or in polynomial time if the geometric representation is given. We partition the vertices of of $G - S$ into sets V_1, V_2, such that $V_1, V_2 \leqslant c' \cdot n$ (for a constant c') and S separates V_1 from V_2. We guess the set $I \subseteq S$ of vertices covered by a fixed optimal solution, and for each vertex $v \in I$ we consider separately cases when the edge covering v belongs to $G_1 := G[V_1 \cup S]$ or to $G_2 := G[V_2 \cup S]$ and solve the problem independently in G_1 and G_2. If a vertex v is chosen to be covered by an edge from, say, G_1, then in the recursive call in G_2 we can remove v and all its neighbors. This gives at most $2 \cdot 3^{|S|} = 2^{\tilde{O}(n^{2/3})}$ recursive calls and the total complexity of this step is $2^{\tilde{O}(n^{2/3})}$.

In the other case, if the number of edges is at least $c/3 \cdot n^{4/3} \log n$, then by Theorem 5, G contains a copy of the biclique $K_{n^{1/3},n^{1/3}}$ as a (non-necessarily induced) subgraph. Let A and B be the bipartition classes of the biclique and consider some fixed solution S. Observe that if $S \cap (A \cup B) \neq \emptyset$, then either (a) S contains an edge uv where $u \in A$ and $v \in B$, and no other vertex of $A \cup B$ is covered, or (b) $V(S)$ contains a vertex u from, say, A, and no vertex of B (or the other way around). In case (a) the edge uv can be chosen in at most $n^{2/3}$

ways and we can safely remove all vertices of the biclique, and in case (b) the vertex u can be chosen in at most $2n^{1/3}$ ways and we remove all vertices of the other bipartition class. Finally, if $S \cap (A \cup B) = \emptyset$, we remove all vertices of the biclique. Thus we obtain a branching algorithm, whose complexity is given by a recursion:

$$\begin{aligned} F(n) \leqslant & 2^{\tilde{O}(n^{2/3})} + \underbrace{F(n-2n^{1/3})}_{S\cap(A\cup B)=\emptyset} + \underbrace{n^{2/3}F(n-2n^{1/3})}_{\text{case (a)}} + \underbrace{2n^{1/3}F(n-n^{1/3})}_{\text{case (b)}} \\ \leqslant & 2^{\tilde{O}(n^{2/3})} + 2n^{2/3}F(n-n^{1/3}). \end{aligned}$$

Note that in case (b) we exhaustively guess the vertex v from $V(S)$ and add it to X in the recursive call. The inequality above is solved in a way analogous to inequality (1). The complexity of this step is again bounded by $2^{\tilde{O}(n^{2/3})}$, hence the running time of the whole algorithm. $\square$

We observe that the approach of Theorem 7 can be further generalized to find induced packings of copies of any fixed graph H. In the Maximum Induced H-Packing problem we are given a graph G and ask for a maximum subset X of vertices of G, such that $G[X]$ consists of disjoint copies of H. Observe that if $H = K_2$, then Maximum Induced H-Packing is precisely Maximum Induced Matching. Even more generally, we can consider a problem of Maximum Induced $\mathcal{H}$-Packing for a fixed finite set of graphs $\mathcal{H}$. In this problem we are given a graph G, and we look for the largest set $X \subseteq V(G)$, such that each connected component of $G[X]$ is isomorphic to some graph from $\mathcal{H}$.

The general idea is exactly the same as in the proof of Theorem 7, but the technical details are significantly more complicated. Recall that in the Maximum Induced Matching problem we needed to exhaustively guess the intersection I of an optimal solution X with the balanced separator S, and for each $v \in I$ we considered the cases that v is matched to a vertex of G_1 or to a vertex of G_2. Now, for each such vertex v, we need to consider the specific graph $H \in \mathcal{H}$, such that the connected component of $G[X]$ containing v is isomorphic to H. Moreover, we need to consider the position of v in the copy of H, and a partition of the remaining vertices of H between G_1 and G_2. However, since $\mathcal{H}$ is a fixed family of fixed graphs, the total number of recursive calls is still $2^{\tilde{O}(n^{2/3})}$. We skip the proof, as it does not bring any new insight and is technically involved.

Theorem 8 Maximum Induced $\mathcal{H}$-Packing *on string graphs with n vertices can be solved in time $2^{\tilde{O}(n^{2/3})}$.* $\square$

Finally, note that if $\mathcal{H}$ contains graphs with different numbers of vertices, then instead of maximizing the number of vertices in X, we can alternatively maximize the number of connected components of $G[X]$.

3 Lower bounds

Rather surprisingly, the win-win for 3-Coloring abruptly ceases to work for k-Coloring for every $k \geqslant 4$. First, let us consider the List 4-Coloring. Following Kratochvíl and Matoušek [31], by Pure 2-Dir we denote graphs admitting a 2-Dir representation in which parallel segments do not intersect. Observe that such a graph is bipartite.

Theorem 9 List *4*-Coloring *of a* Pure 2-Dir *graph cannot be solved in time* $2^{o(n)}$*, even if each list has size at most 3, unless the ETH fails.*

Proof. Let Φ be a 3-Sat formula with n variables $v_1, v_2, \dots, v_n$ and m clauses $C_1, C_2, \dots, C_m$. By repeating some literals in a clause, we may assume that each clause contains exactly three literals. For a clause C_i, let v^i_1, v^i_2, v^i_3 denote the variables of C_i.

We construct a 2-Dir graph G with lists L of colors from $\{1, 2, 3, 4\}$, such that Φ is satisfiable if and only if G is list-colorable with respect to the lists L.

For each variable v_i, we introduce a horizontal segment called x_i. For each clause C_i we introduce three vertical segments y^i_1, y^i_2, y^i_3, corresponding to v^i_1, v^i_2, and v^i_3, respectively. We arrange them in a grid-like way (see Fig. 1). One may observe that the intersection graph induced by those segments is a biclique. We set the lists of each x_i to $\{1, 2\}$ and the lists of each y^i_1, y^i_2, y^i_3 to $\{3, 4\}$.

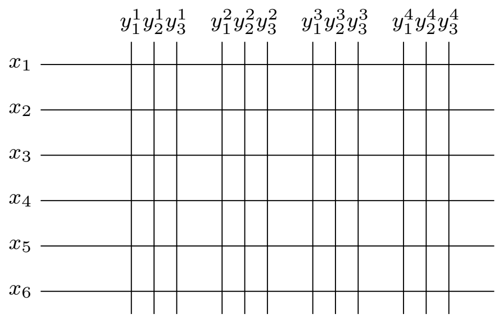
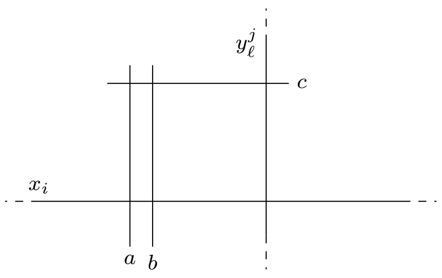


Fig. 1 The arrangement of variable- and occurrence-segments in G.

The colors 1 and 2 used for x_i will be interpreted, respectively, as *true* and *false* values given to v_i, while the colors 3 and 4 given to y^i_j will be interpreted, respectively, as *true* and *false* values given to the literal corresponding to v^i_j.

To ensure this, we need to introduce *equality gadgets* and *inequality gadgets.* If the variable v_i appears positively in the clause C_j as its ℓ-th literal, then at the crossing point of x_i and y^j_ℓ we put the equality gadget ensuring that in any feasible coloring of G, the color of x_i is 1 (2, respectively) if and only if the color of y^i_ℓ is 3 (4, respectively). On the other hand, if v_i appears negatively in C_j as its ℓ-th literal, then at the crossing point of x_i and y^j_ℓ we put the inequality gadget ensuring that in any feasible coloring of G, the color of x_i is 1 (2, respectively) if and only if the color of y^j_ℓ is 4 (3, respectively).

The equality gadget consists of 3 segments, arranged as depicted on Fig. 2 and uses lists from the middle table on Fig. 2. Suppose x_i gets the color 1. Then a receives color 3, and c gets the color 4. Thus the only choice for the color for y^j_ℓ is 3. The coloring can be extended by coloring b to 2. The other cases are symmetric. The inequality gadget is analogous and uses the lists on the right side of Fig. 2.



vertex	list
x_i	1,2
y^j_ℓ	3,4
a	1,3
b	2,4
c	3,4

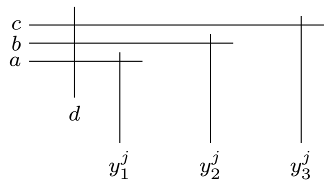
Lists in the equality gadget.

vertex	list
x_i	1,2
y^j_ℓ	3,4
a	1,4
b	2,3
c	3,4

Lists in the inequality gadget.

Fig. 2 Equality and inequality gadgets. The arrangement of segments is the same in both gadgets, the only difference is the lists.

The only thing left is to ensure that the coloring of y^j_1, y^j_2, y^j_3 exists if and only if C_j is satisfied. This is ensured by the *satisfiability gadget* depicted in Fig. 3, attached to the top ends of y^1_j, y^j_2, and y^j_3. Note that the gadget can be colored if and only if at least one of y^j_1, y^j_2, y^j_3 gets color 3, which is equivalent to one literal of C_j being set to *true*.



vertex	list
y^j_ℓ	3,4
a	1,4
b	2,4
c	3,4
d	1,2,3

Fig. 3 Satisfiability gadget.

The number of vertices of G is $n' = \underbrace{n}_{x_i} + \underbrace{3m}_{y^j_\ell} + \underbrace{9m}_{\text{(in)equality}} + \underbrace{4m}_{\text{satisfiability}} = \Theta(n+m)$. On the other hand, an algorithm solving list coloring of G in time $2^{o(n')}$ can be used to decide the satisfiability of Φ in time $2^{o(n')} = 2^{o(n+m)}$, which in turn contradicts the ETH. □

The non-list version is obtained analogously to the hardness for 6-Coloring in [5]. We include the proof to make the paper self-contained.

Theorem 10 *For every fixed $k \geqslant 4$, the k-*Coloring *problem of a* 2-Dir *graph cannot be solved in time $2^{o(n)}$, unless the ETH fails.*

Proof of Theorem 10. We modify the construction from the proof of Theorem 9. We first introduce k overlapping segments $R_1, R_2, \ldots, R_k$, whose coloring will serve as a reference coloring. Since these segments are pairwise intersecting, each of them receives a different color. We will denote by $i \in [k]$ the color assigned to R_i.

Now, for each segment v of G, we want to simulate the list $L(v)$ from the instance of List 4-Coloring constructed in the proof of Theorem 9. For every color $i \notin L(v)$, we want to introduce a segment s_i intersecting v, which will always receive color i.

To achieve this, we first need to transport the reference coloring to every gadget. We split it into two parts – we will separately transport colors 1 and 2, and colors greater than 2. The overall high-level idea is depicted in Figure 4. Observe that this already simulates the lists for every x_i.

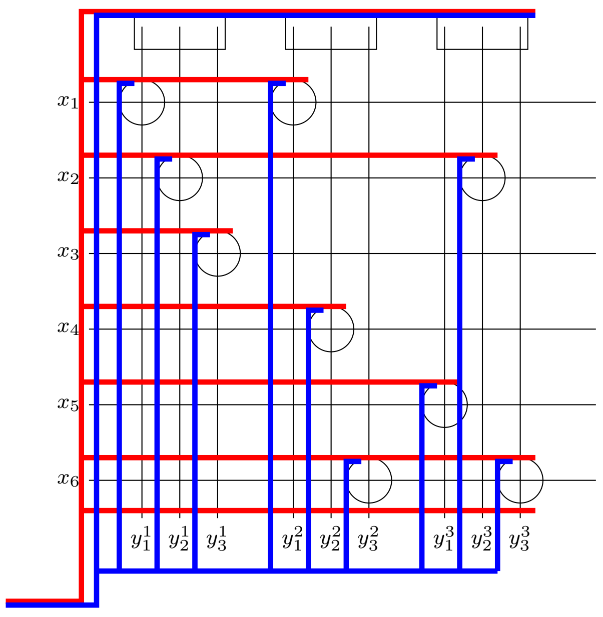


Fig. 4 Reference coloring is transported to every gadget. Circles denote the (in)equality gadgets, while rectangles denote the satisfiability gadgets. Red and blue lines denote, respectively, pairs of overlapping segments with colors 1,2, and colors greater than 2. Segments $R_1, R_2, \ldots, R_k$ are positioned in the lower left corner of the picture.

Such a construction relies on a constant-size gadget, which allows us to turn or split the reference coloring. The construction of this gadget is depicted in Figure 5. Note that the number of segments in this gadget is constant if k is constant. Moreover, turning or splitting the reference coloring of fewer than k colors can be obtained by a simple adaptation of the turning gadget. Indeed, suppose we want to introduce a turning gadget for the set of colors $C \subseteq [k]$, with $|C| = k' < k$ and we have k' overlapping segments carrying these colors. We introduce $k - k'$ dummy segments, overlapping these segments. The dummy segments will clearly receive colors from $[k] \setminus C$, but we do not know which segment will get which color. Now we introduce a turning gadget for k colors. We know which segments leaving the turning gadget get colors in C (and we precisely know which segment gets which color). We do not need the remaining segments anymore, so we can finish them as soon as they leave the turning gadget.

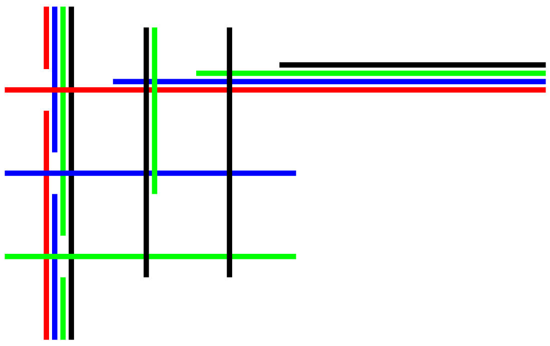


Fig. 5 Turning gadget for $k = 4$ colors. The parallel segments depicted close to each other are overlapping. Observe that the depicted 4-coloring is the only possible (up to the permutation of colors). For $k > 4$ the turning gadget is analogous.

Now, the only thing left is to connect every segment in every gadget to appropriate segments carrying the reference coloring (note that each y^j_ℓ belongs to some (in)equality gadget). This can easily be done using a constant number of additional segments per gadget: we introduce a turning gadget for k colors, and finish the segments that are not needed anymore before they intersect the segments in gadgets (see Figure 6).

The total size of the construction increases by a constant factor, as we introduce $O(n)$ constant-size turning gadgets. Thus an algorithm for k-coloring the constructed 2-Dir graph in time $2^{o(n')}$ could be used to solve any 3-Sat instance in time $2^{o(n)}$, refuting the ETH. □

Observe that the construction in the proof of Theorem 9 cannot be performed with segments of bounded lengths, since segments x_i and y^j_k need to have length $O(n)$ (while the segments inside the gadgets can have unit length). For unit segments, we show the following weaker lower bound.

Theorem 11 *For every fixed $k \geqslant 4$,* List k-Coloring *of a unit* 2-Dir *graph or k-*Coloring *of a unit* 3-Dir *cannot be solved in time $2^{o(n^{2/3})}$, unless the ETH fails.*

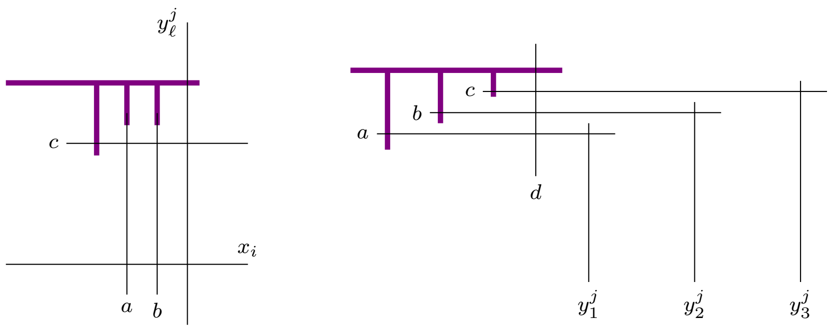


Fig. 6 Simulation of lists for vertices in (in)equality gadgets and satisfiability gadgets. Horizontal violet lines denote tuples of overlapping segments, carrying the reference coloring (all k colors). In the vertical lines, segments carrying unnecessary colors are finished before they intersect the segments of the gadgets.

Proof. Consider a 3-Sat instance with variables $v_1, v_2, \ldots, v_n$ and clauses $C_1, C_2, \ldots, C_m$, where $m = \Theta(n)$. By duplicating some literals if necessary, we may assume that each clause contains exactly three literals.

Let us start with a reduction to List k-Coloring in unit 2-Dir. For each clause C_j we introduce three disjoint vertical unit segments, each corresponding to one literal in C_j. These segments will be called *literal segments related to* C_j. We place them in such a way that the distance between the leftmost and the rightmost literal segment (of all literal segments) is slightly smaller than $1/2$. The ordering of the segments is the following: first, the literal segments related to the clause C_1, then literal segments related to the clause C_2 and so on. Moreover, they are slightly shifted vertically, so that the y-coordinates of their top endpoints form an increasing sequence. We set the list of possible colors for each literal segment to $\{3, 4\}$ and for each three literal segments related to a single clause, we introduce a satisfiability gadget already shown in Figure 3. We will interpret 3 as assigning the value *true* to the particular literal (and 4 will correspond to *false*). Figure 7 shows the placement of literal segments and satisfiability gadgets. Analogously, for each variable v, we

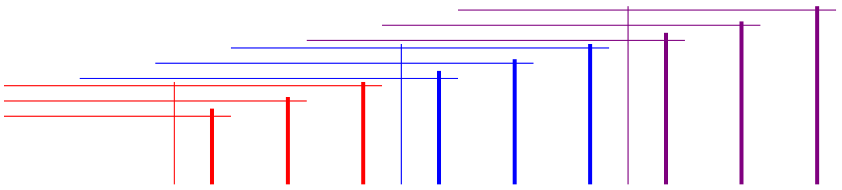
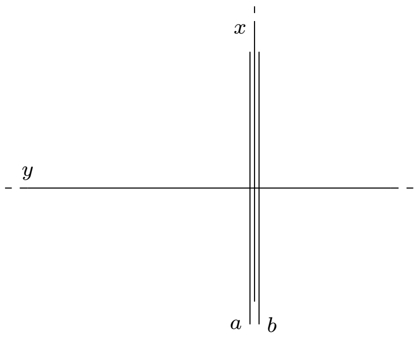


Fig. 7 Placement of satisfiability gadgets. Segments in one color correspond to one clause. Literal segments are depicted by thick lines. Note that all segments can be freely extended to the left and to the bottom, to make them unit.

introduce a vertical segment for each occurrence of v (we call these segments *occurrence segments*) with list $\{3, 4\}$. The segments are placed in such a way that the distance between the leftmost and the rightmost occurrence segment is slightly smaller than $1/2$. Moreover, leftmost occurence segments correspond to v_1, then we put the segments for v_2 etc. For a variable v_i for $i \in [n]$, we introduce a horizontal segment, intersecting the occurrence segments of variables $v_i, v_{i+1}, \ldots, v_n$. These segments will be called *variable segments*. The variable segments are pairwise disjoint and each has list $\{1, 2\}$. Again, we will interpret 1 as the value *true* given to a variable, and 2 will denote *false*. Now we need to ensure that the truth assignment defined by the coloring of occurrence segments is consistent. For this, we will use a slightly modified version of the (in)equality segment introduced in Figure 2. The modified gadget is shown in Figure 8. For a variable v and its positive occurrence, we introduce an equality



vertex	list
x	3,4
y	1,2
a	3,2
b	4,1

Lists in the equality gadget.

vertex	list
x	3,4
y	1,2
a	3,1
b	4,2

Lists in the inequality gadget.

Fig. 8 Modified equality and inequality gadgets. The segment x is an occurrence segment and the segment y is a variable segment. Segments a, b, and x overlap.

gadget joining the occurrence segment and the variable segment. Analogously, we introduce an inequality gadget joining the variable segment and the occurrence segment corresponding to a negative occurrence. The placement of all these segments is shown in Figure 9.

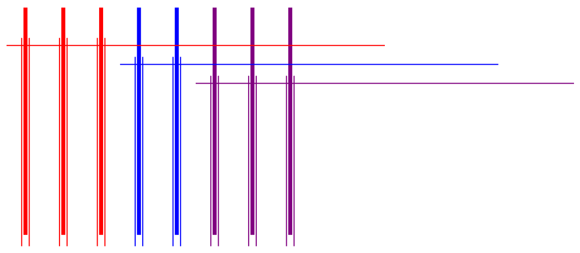


Fig. 9 Placement of variable and occurrence segments along with (in)equality gadgets. Segments in one color correspond to one variable. Literal segments are depicted by thick lines. Thin horizontal lines are variable segments and thin vertical lines are parts of (in)equality gadgets.

Now we need to make sure that the truth assignment given by the coloring of literal segments is consistent with the truth assignment given by the coloring of occurrence segments. We will do it in a very similar way as we synchronized the colorings of occurrence segments, i.e., by using auxiliary horizontal segments and equality gadgets. Let $\ell_1, \ell_2, \ldots, \ell_{3m}$ denote the literals ordered as their corresponding literal segments (from left to right). Let $o_1, o_2, \ldots, o_{3m}$ be the ordering of occurrences, again from left to right. Let σ be the permutation of $[3m]$, such that the literal ℓ_i corresponds to the occurrence $o_{\sigma(i)}$. Now, for every $i \in [n]$, we want to introduce an equality gadget between the literal segment corresponding to ℓ_i and the occurrence segment corresponding to $o_{\sigma(i)}$.

We observe that this is quite easy to do if σ is either increasing or decreasing. It follows from basic properties of Young tableaux (see Knuth [28, Sec. 5.1.4]) that each permutation of $[3m]$ can be partitioned into $z \leqslant \lceil \sqrt{6m + 1/4} - 1/2 \rceil = O(\sqrt{n})$ monotone sequences, and this partition can be found in polynomial time (see also Brandstädt and Kratsch [10]). Let $\sigma_1, \sigma_2, \ldots, \sigma_z$ be the partition of σ, where each σ_i is monotone. We introduce z *layers*, each corresponding to one σ_i. The ith layer is responsible for synchronizing the colorings of literal segments and occurrence segments corresponding to elements of σ_i (we call such segments *important* for layer i). Figure 10 shows a single layer, we set the distance between horizontal segments to be very small, compared to the length of each segment. Moreover, the horizontal segments cross the vertical literal and occurrence segments near their middle points. Note that each layer contains copies of *all* literal segments and occurrence segments. They appear in two groups – literal segments on the left and occurrence segments on the right. The distance between leftmost and rightmost segment in one group is slightly less than $1/2$, and the distance between the rightmost literal segment and the leftmost occurrence segment is slightly less than 2. This allows us to fit two equality gadgets, whose horizontal segments are collinear but non-overlapping (see Fig. 10 and notice that we can adjust the distances within groups and between the groups, so that the distance between the horizontal segments is always positive and smaller than 1). For each such pair we introduce $k-1$ overlapping segments with lists $\{1, 2, \ldots, k\}$, intersecting both of them and nothing else.

The last thing to do is to connect the literal segments (with (in)equality gadgets), layers, and occurrence segments with satisfiability gadgets. We place the occurrence segments at the bottom, and then we introduce layers in such a way that the corresponding vertical segments are collinear but non-intersecting. Finally, the literal segments with satisfiability gadgets are put on top (see Fig. 12 left).

Now consider two consecutive layers i and $i+1$. Let s_1 and s_2 be collinear vertical segments and let s_1 be above s_2 (i.e., s_1 is in the layer $i+1$, and s_2 is in the layer i, see Fig. 11). Let us assume that σ_i is increasing, the other case is symmetric. Observe that the top end of s_2 is not covered by (i.e., does not belong to) any other segment introduced so far. If s_1 is an important literal segment of the ith layer, then there are two segments a, b belonging to the

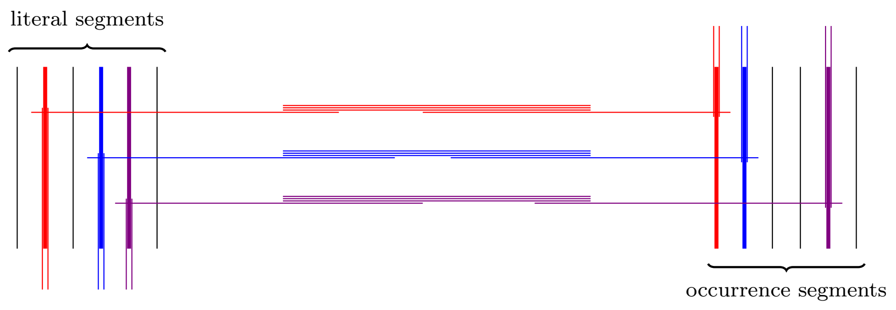


Fig. 10 A single layer i. Thick colored segments indicate corresponding literal and occurrence segments, which are important for layer i. Thin colored segments are parts of equality gadgets.

appropriate equality gadget, which are covering the lower endpoint of s_1. We introduce $k-3$ overlapping vertical segments $q_1, q_2, \ldots, q_{k-3}$, intersecting only s_1, a, b (so its upper endpoint is below the horizontal segment of the equality gadget, where a and b belong), and s_2, and set their lists to $\{1, 2, \ldots, k\}$. The distance between the layers can be adjusted do that these newly introduced segments have no other intersections. Indeed, this is possible since the horizontal segments in each layer cross the vertical ones near their middle points.

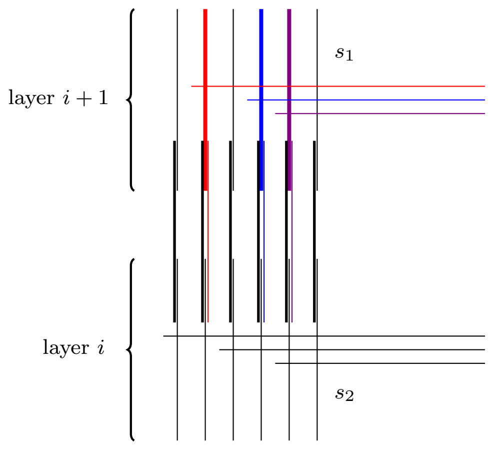
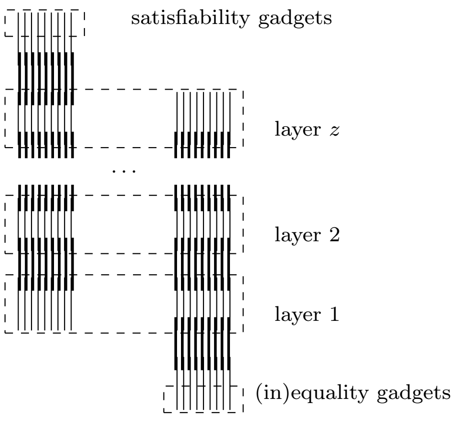


Fig. 11 Literal segments of two consecutive layers i and $i+1$. Thick colored segments indicate segments that are important in the given layer. Thin colored segments are parts of equality gadgets (they denote overlapping segments a and b. Thick black segments are bundles of segments q_i.

If s_1 is not an important segment (and s_2 is important or not), we introduce $k-1$ overlapping segments $q_1, q_2, \ldots, q_{k-1}$ with lists $\{1, 2, \ldots, k\}$, intersecting s_1, s_2, and no other previously constructed segment. This way s_1 and s_2 are non-adjacent, but they are both intersected by $k-1$ pairwise intersecting segments. This ensures that s_1 and s_2 will receive the same color.

In an analogous way we synchronize the colorings of occurrence segments, however this time segments from equality gadgets cover top ends of important occurrence segments.



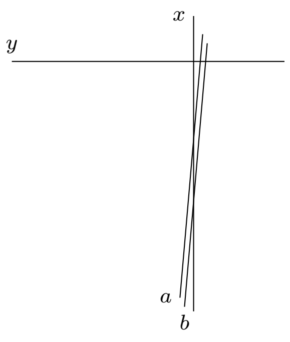


Fig. 12 Left: Overall construction. Right: Modified equality gadgets in a unit 3-Dir graph.

The number of segments in each layer is $O(n)$ and the number of layers is $z = O(\sqrt{n})$. Thus the total number of segments in our construction is $O(n^{3/2})$. This implies that an algorithm solving List k-Coloring on unit 2-Dir graphs with N vertices in time $2^{o(N^{2/3})}$ could be used to solve 3-Sat with n variables in time $2^{o(n)}$, which contradicts the ETH.

If we want to obtain a reduction to the non-list k-Coloring for any $k \geqslant 4$, we need to transport the reference coloring to each gadget. However, this cannot be done for segments a, b in our (in)equality gadgets (recall Theorem 10 and Figure 8), which have non-trivial lists and are fully covered by other segments – note that if a segment in color 4 intersects a, it will also intersect x (note that the segments with lists $\{1, 2, \dots, k\}$, even if they are fully covered, are not problematic since they do not need the reference coloring). But if we use a third direction, we can make a, b intersect x and y (again, we use notation in Figure 8) and no other vertex with non-trivial list in our grid-like structure – it is enough to choose their slope to be "almost vertical" (see Fig. 12, right). Now we can transport the reference coloring just as we did in the proof of Theorem 10, see Fig. 4. This construction adds a linear number of segments per layer, which is $O(n^{3/2})$ segments in total.

This shows the claimed lower bound, for every fixed $k \geqslant 4$, for k-Coloring of unit 3-Dir graphs and completes the proof. □

We show that on segment graphs, Min Dominating Set, Min Connected Dominating Set, and Min Independent Dominating Set are unlikely to have a subexponential algorithm.

Theorem 12 Min (Connected) Dominating Set *cannot be solved in time* $2^{o(n)}$ *on segment graphs with n vertices, unless the ETH fails.*

Proof. By introducing new variables and clauses, we can transform an arbitrary 3-Sat formula ψ with N' variables and $M' = O(N')$ clauses into an equivalent Cnf-Sat formula ϕ with $N = O(N' + M')$ variables and $M = O(N' + M')$ clauses, where each variable appears exactly twice positively and twice negatively. Indeed, say a variable x_i appears t times, one can create t copies $x_i^1, x_i^2, \dots, x_i^t$ and equate them by the chain of implications $\neg x_i^1 \vee x_i^2, \neg x_i^2 \vee x_i^3, \dots \neg x_i^{t-1} \vee x_i^t, \neg x_i^t \vee x_i^1$. Now, each occurrence of x_i is replaced by a different x_i^s. To use all the copies of the chain of implications exactly once more positively and negatively, one can add equivalent literals to a clause initially containing x_i or $\neg x_i$.

Put M pairwise-disjoint small segments on a circle as shown in Figure 13; M slightly perturbed points work as long as no two pairs define the same direction. Each small segment $s(C_j)$ represents a distinct clause C_j. For each literal σx_i, where x_i is one of the N variables appearing in ϕ and $\sigma \in \{\emptyset, \neg\}$, we add a segment $s(\sigma x_i)$ crossing only the two small segments $s(C_j)$ and $s(C_{j'})$ corresponding to the clauses this literal satisfies. We extend all the segments corresponding to literals to make them pairwise intersect.

For each pair of literals $x_i, \neg x_i$, we add a small segment $s(i)$ near the intersection of $s(x_i)$ and $s(\neg x_i)$ which intersects only $s(x_i)$ and $s(\neg x_i)$. This finishes the construction (see Figure 13). Note that the total number of segments is $n := 3N + M = \Theta(N)$. We claim that there is a dominating set of size N in the intersection graph if and only if ϕ is satisfiable.

Indeed, to dominate all the segments $s(i)$ for $i \in [N]$, one has to take at least one of $s(x_i)$ and $s(\neg x_i)$, hence exactly one of them since the sought after dominating set should have size N. The choice of the literals will only dominate all the segments $s(C_j)$ for $j \in [M]$ if the chosen dominating set corresponds to a satisfying assignment. The reverse direction is straightforward.

Moreover, as every pair of segments representing literals intersects, the dominating set encoding the satisfiable assignment is connected (it even induces a clique). □

The hardness proof for Min Independent Dominating Set has to be quite different and is trickier.

Theorem 13 Min Independent Dominating Set *cannot be solved in time* $2^{o(n)}$ *on segment graphs with n vertices, unless the ETH fails.*

Proof. We reduce from 3-SAT on N variables and M clauses, where each literal appears at most four times, and each clause contains *exactly* three literals. This restriction was shown NP-complete by Tovey [45]. We can further assume that each *literal* appears at most three times. Indeed, if the same literal appears four times, then by definition its negation cannot appear in the whole instance. Hence, the variable can be set so as to satisfy the four corresponding clauses.

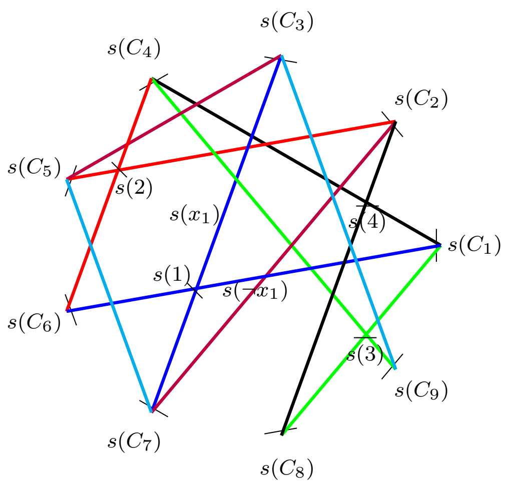


Fig. 13 An example with 9 clauses and 6 variables. The segments $s(x_i)$ and $s(\neg x_i)$ can be inferred from $s(i)$ or from the colors (although we only specified which is which for variable x_1). Observe that the light blue and purple pairs do not intersect inside the circle so those segments should be extended outside it until they meet (this part is not shown in the picture).

The variable gadget $G(x_i)$ for variable x_i consists of three parallel segments T_i representing positive occurrences crossing three parallel segments F_i representing negative occurrences. Those two sets of segments intersect three dummy pairs of parallel segments as shown on Figure 14. Note that even if a literal appears strictly fewer than three times, we keep exactly three parallel segments to encode it.

The clause gadget $G(C_j)$ for the clause C_j consists of three pairwise crossing segments, drawn in blue in Figure 14, each of which crosses one red segment. Each pair of blue and red segment corresponds to one of literals of C_j, these segments will be called *literal segments*. Additionally, all blue segments are intersected by four parallel dummy segments, and each pair of crossing blue and red segment has a private segment, crossing both of them.

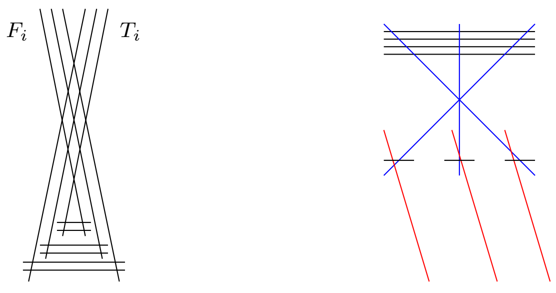


Fig. 14 The variable gadget $G(x_i)$ for the variable x_i (left) and the clause gadget $G(C_j)$ for the clause C_j (right).

Now, for every literal x_i $(\neg x_i)$ belonging to the clause C_j, we add a new segment denoted by $s(x_i, C_j)$ or $s(\neg x_i, C_j)$, respectively. We call these segments *occurrence segments*. This segment crosses one non-dummy segment in $G(x_i)$: a segment of T_i if the literal is positive, and a segment of F_i otherwise. Moreover, it crosses one literal (red) segment from $G(C_j)$, and no other segments in variable and clause gadgets, see Figure 15. Each red literal segment is crossed by exactly one occurrence segment, and each segment of $\bigcup_{i \in [n]} T_i \cup F_i$ is crossed by at most one occurrence segment.

We claim that such a constructed graph has an independent dominating set of size at most $3N + 3M$ if and only if the initial formula is satisfiable.

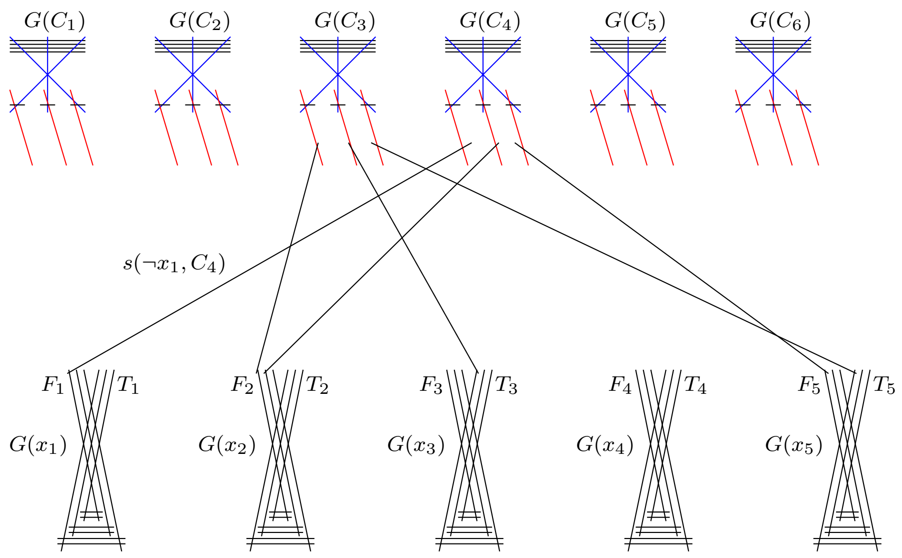


Fig. 15 The overall picture. We only represented two clauses: $C_3 = \neg x_2 \vee x_3 \vee x_5$ and $C_4 = \neg x_1 \vee \neg x_2 \vee \neg x_5$.

First, suppose that A is a satisfying assignment. If x_i is set to true by A, we select the three segments of T_i in the solution, otherwise we select the three segments of F_i. Now consider a clause C_j. Since A is satisfying, it contains at least one true literal. In $G(C_j)$ we select the blue segment corresponding to a true literal and the red segments corresponding to the other two literals. Note that this way we select $3N + 3M$ segments and the selected set dominates all segments from all gadgets. Let us consider an occurrence segment $s((\neg)x_i, C_j)$. Either it is dominated by one of the selected red segments in $G(C_j)$, or it corresponds to a true literal of C_j, so it is dominated by a selected segment in $G(x_i)$.

On the other hand, assume that there is an independent dominating set S of size at most $3N + 3M$. Notice that in order to dominate all dummy segments in $G(x_i)$, we need to select at least three segments from $G(x_i)$, and if we want to select exactly three, we need to choose either all segments in T_i, or all segments in F_i. Analogously, we need to select at least three segments from each $G(C_j)$, and if we want to select exactly three, we need do choose one of blue segments and thus we cannot choose its corresponding red segment. Note that since the total size of S is $3N + 3M$, we need to select exactly three segments in each gadget, and no occurrence segment is selected.

We define the assignment A as follows: if segments T_i are in S, then x_i is set true, and otherwise x_i is set false. Suppose that C_j is not satisfied by A, which means that all its literals are false. This means that the three occurrence segments joining appropriate variable gadgets with $G(C_j)$ are not dominated by segments in variable gadgets, so S must contain all three red segments from $G(C_j)$. However, this way the horizontal segments from $G(C_j)$ are not dominated, a contradiction.

The total number of segments is bounded by $12N+13M+3M = O(N+M)$, so the claim holds. □

Theorem 14 Max Clique *cannot be solved in time* $2^{o(n)}$ *on string graphs with* n *vertices, unless the ETH fails.*

Proof. We reduce from 3-Sat with a linear number of clauses, where every clause contains exactly three literals. Let ϕ be an instance with N variables and $M = \Theta(N)$ clauses. For any positive integers p and s, the *co-cluster* $\text{CC}_{p,s} = K_{s,s,\ldots,s \text{ (p times)}}$ can be represented as in Figure 16. We encode the

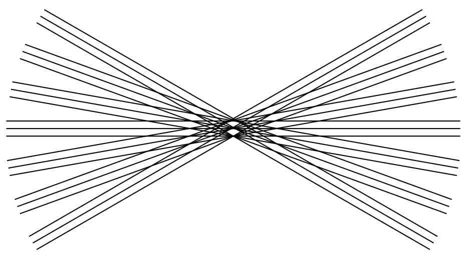


Fig. 16 Realization of a co-cluster $\text{CC}_{p,s} = K_{s,s,\ldots,s}$ with $s = 3$ and $p = 7$.

N variables by $2N$ curves representing *true* and *false* for each variable by a co-cluster $\text{CC}_{N,2}$ and the $M = \Theta(N)$ clauses by $3M$ curves each representing a distinct literal in a clause by a co-cluster $\text{CC}_{M,3}$. There are in total $n := 2N + 3M = \Theta(N)$ curves. We make the $2N$ *variable* curves intersect the $3M$ *literal* curves in a grid-like way. They form an almost complete biclique $K_{2N,3M}$ where $3M$ edges are removed.

More precisely, the *literal* curve $c(l_i^j)$ ($j \in [M]$, $i \in [3]$) intersects every *variable* curve but $c(\sigma x_k)$ ($k \in [N]$) encoding the k-th variable with sign $\sigma \in \{\emptyset, \neg\}$ for which l_i^j and σw_k are opposite literals (see Figure 17). It is easy to observe that there is a clique of size $N + M$ if and only if ϕ has a satisfying assignment. □

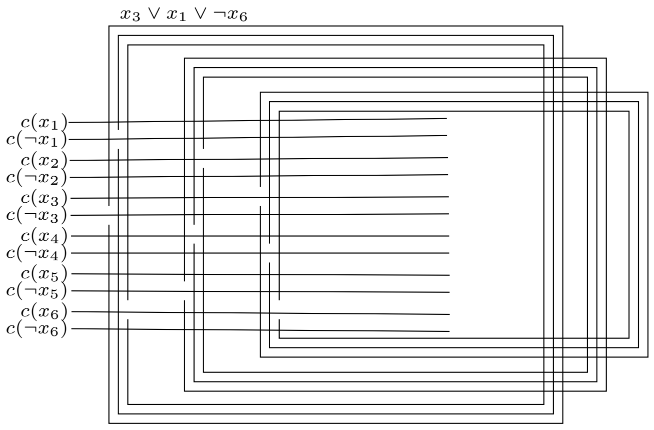


Fig. 17 The representation of 3 clauses: $x_3 \vee x_1 \vee \neg x_6$, $x_5 \vee \neg x_4 \vee \neg x_2$, and $\neg x_3 \vee x_4 \vee \neg x_6$.

4 Perspectives

We have started a generalized optimality program on segment and string graphs for the most principal graph problems. On the algorithmic side, we extended a subexponential algorithm for Max Independent Set on string graphs [21] to some other problems, namely 3-Coloring, Min Feedback Vertex Set, and Maximum Induced Matching. On the complexity side, we showed that no subexponential algorithm is likely for, among others, 4-Coloring, Min Dominating Set, and Min Independent Dominating Set. It is quite easy to obtain such lower bounds for string graphs. Extending those results to segments requires more ingenuity, and even more so when it comes to unit segments.

A handful of questions remains unsettled. Can we improve the algorithm or give tight ETH lower bounds for the following problems: Max Independent Set without geometric representation, 3-Coloring, and Min Feedback Vertex Set on segments/strings? Can we show for Max Clique the same lower bound for segment graphs as we have for string graphs. The mere NP-hardness of Max Clique on segments [11] answered a long-standing open question. Hence, it is likely that getting a tight ETH hardness will be difficult. We would also find interesting to have, for a certain problem, an algorithm for segments (resp. unit segments) which beats the ETH lower bound on strings (resp. segments). So far, we only have candidate problems for such a *"separation"*.

Finally, another natural continuation of this work is to determine which fixed-parameter tractable problems can be solved in time $O^*(2^{\tilde{O}(k^{2/3})})$ or $O^*(2^{\tilde{O}(\sqrt{k})})$, and which W[1]-hard problems can be solved in time $f(k)n^{O(\sqrt{k})}$ on segments and strings. For instance, Min Vertex Cover can be solved in time $O^*(2^{\tilde{O}(k^{2/3})})$ (even in time $O^*(2^{\tilde{O}(\sqrt{k})})$ if a geometric representation is given with $O^*(2^{\tilde{O}(\sqrt{k})})$ intersections) on string graphs due to the linear kernel yielding an equivalent instance on $2k$ vertices and the algorithm for Max Independent Set. The latter problem can be solved in $n^{O(\sqrt{k})}$ in segments or more generally in polygons of polynomial complexity [35], while Min Dominating Set on string graphs cannot be solved in time $f(k)n^{o(k)}$, for any computable function f, unless the ETH fails (since this lower bound holds for split graphs).

References

1. A. Adamaszek, S. Har-Peled, and A. Wiese. Approximation schemes for independent set and sparse subsets of polygons. *CoRR*, abs/1703.04758, 2017.
2. A. Adamaszek and A. Wiese. A QPTAS for maximum weight independent set of polygons with polylogarithmically many vertices. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 645–656, 2014.
3. J. Alber and J. Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004.
4. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, Jan. 1994.
5. C. Biró, É. Bonnet, D. Marx, T. Miltzow, and P. Rzążewski. Fine-grained complexity of coloring unit disks and balls. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 18:1–18:16, 2017.
6. É. Bonnet, N. Brettell, O. Kwon, and D. Marx. Generalized feedback vertex set problems on bounded-treewidth graphs: Chordality is the key to single-exponential parameterized algorithms. In D. Lokshtanov and N. Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 7:1–7:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
7. É. Bonnet, N. Brettell, O. Kwon, and D. Marx. Generalized feedback vertex set problems on bounded-treewidth graphs: chordality is the key to single-exponential parameterized algorithms. *CoRR*, abs/1704.06757, 2017.
8. É. Bonnet and P. Rzążewski. Optimality program in segment and string graphs. In A. Brandstädt, E. Köhler, and K. Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 79–90. Springer, 2018.
9. J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *J. Graph Algorithms Appl.*, 8(2):241–273, 2004.

10. A. Brandstädt and D. Kratsch. On partitions of permutations into increasing and decreasing subsequences. *Elektronische Informationsverarbeitung und Kybernetik*, 22(5/6):263–273, 1986.
11. S. Cabello, J. Cardinal, and S. Langerman. The clique problem in ray intersection graphs. *Discrete & Computational Geometry*, 50(3):771–783, 2013.
12. J. Chalopin and D. Gonçalves. Every planar graph is the intersection graph of segments in the plane: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 631–638, 2009.
13. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004.
14. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.
15. E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.
16. E. D. Demaine and M. Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.
17. E. D. Demaine and M. T. Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *GD 2014 Proc.*, pages 517–533, 2004.
18. P. Erdős and G. Szekeres. *A Combinatorial Problem in Geometry*, pages 49–56. Birkhäuser Boston, Boston, MA, 1987.
19. F. V. Fomin, D. Lokshtanov, F. Panolan, S. Saurabh, and M. Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *CoRR*, abs/1704.07279, 2017.
20. F. V. Fomin, D. Lokshtanov, and S. Saurabh. Bidimensionality and geometric graphs. In Y. Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1563–1575. SIAM, 2012.
21. J. Fox and J. Pach. Computing the independence number of intersection graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1161–1165, 2011.
22. J. Fox and J. Pach. Applications of a new separator theorem for string graphs. *CoRR*, abs/1302.7228, 2013.
23. J. Fox, J. Pach, and C. D. Tóth. A bipartite strengthening of the crossing lemma. *J. Comb. Theory, Ser. B*, 100(1):23–35, 2010.
24. D. Gonçalves, L. Isenmann, and C. Pennarun. Planar Graphs as L-intersection or L-contact graphs. In A. Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 172–184. SIAM, 2018.

25. S. Har-Peled. Quasi-polynomial time approximation scheme for sparse subsets of polygons. In *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08 - 11, 2014*, page 120, 2014.
26. R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367 – 375, 2001.
27. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.
28. D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.
29. J. Kratochvíl. String graphs. II. recognizing string graphs is NP-hard. *J. Comb. Theory, Ser. B*, 52(1):67–78, 1991.
30. J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *J. Comb. Theory, Ser. B*, 53(1):1–4, 1991.
31. J. Kratochvíl and J. Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289 – 315, 1994.
32. J. R. Lee. Separators in region intersection graphs. *CoRR*, abs/1608.01612, 2016.
33. R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.
34. D. Marx. Parameterized complexity of independence and domination on geometric graphs. In *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, pages 154–165, 2006.
35. D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In N. Bansal and I. Finocchi, editors, *ESA 2015 Proc.*, volume 9294 of *LNCS*, pages 865–877. Springer, 2015.
36. D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams. *CoRR*, abs/1504.05476, 2015.
37. J. Matoušek. Intersection graphs of segments and $\exists\mathbb{R}$. *CoRR*, abs/1406.2636, 2014.
38. J. Matoušek. Near-optimal separators in string graphs. *Combinatorics, Probability & Computing*, 23(1):135–139, 2014.
39. C. McDiarmid and T. Müller. Integer realizations of disk and segment graphs. *J. Comb. Theory, Ser. B*, 103(1):114–143, 2013.
40. N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994.
41. M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. *J. Comput. Syst. Sci.*, 67(2):365–380, 2003.
42. M. Schaefer and D. Štefankovič. Fixed Points, Nash Equilibria, and the Existential Theory of the Reals. *Theory of Computing Systems*, 60(2):172–193, Feb 2017.
43. E. Scheinerman. *Intersection classes and multiple intersection parameters of graphs*. PhD thesis, Princeton University, 1984.

44. W. D. Smith and N. C. Wormald. Geometric separator theorems and applications. In *FOCS 1998 Proc.*, pages 232–243, Washington, DC, USA, 1998. IEEE Computer Society.
45. C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.
46. I. E. Zverovich and V. E. Zverovich. An induced subgraph characterization of domination perfect graphs. *Journal of Graph Theory*, 20(3):375–395, 1995.