On the size of outer-string representations

Therese Biedl[1]
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
biedl@uwaterloo.ca

Ahmad Biniaz[2]
Cheriton School of Computer Science, University of Waterloo
Waterloo, Canada
ahmad.biniaz@gmail.com

Martin Derka[3]
School of Computer Science, Carleton University
Ottawa, Canada
mderka@uwaterloo.ca

Abstract

Outer-string graphs, i.e., graphs that can be represented as intersection of curves in 2D, all of which end in the outer-face, have recently received much interest, especially since it was shown that the independent set problem can be solved efficiently in such graphs. However, the runtime for the independent set problem depends on N, the number of segments in an outer-string representation, rather than the number n of vertices of the graph. In this paper, we argue that for some outer-string graphs, N must be exponential in n. We also study some special string graphs, viz. monotone string graphs, and argue that for them N can be assumed to be polynomial in n. Finally we give an algorithm for independent set in so-called strip-grounded monotone outer-string graphs that is polynomial in n.

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Graph theory

Keywords and phrases string graph, outer-string graph, size of representation, independent set

Digital Object Identifier 10.4230/LIPIcs.SWAT.2018.10

1 Introduction

A *string graph* is a graph $G = (V, E)$ that has a *string representation*, i.e., an assignment of curves in the plane to the vertices in such a way that two vertices v, w are connected by an edge (v, w) if and only if their corresponding curves $\mathbf{v}, \mathbf{w}$ intersect. In this paper, we only consider string representations where any two curves $\mathbf{v}$ and $\mathbf{w}$ intersect in a finite set of points (denoted $\mathbf{v} \cap \mathbf{w}$). We will always use bold-face $\mathbf{v}$ to denote the curve of a vertex v.

The study of string graphs goes back over 50 years, see e.g. [24, 7]. It is known that every planar graph is a string graph [7], but in general, testing whether a graph is a string graph is NP-complete [15, 20, 22]. Many variants of string graphs have been studied in the literature. Of chief interest to us are the so-called *outer-string* graphs, which have a string representation such that for every vertex v the curve $\mathbf{v}$ has at least one endpoint on

[1] Supported by NSERC.
[2] Supported by NSERC Postdoctoral Fellowship.
[3] Supported by NSERC Vanier fellowship while author was a student at University of Waterloo.

the outer-face of the string representation. See some recent articles [2, 3] for some results concerning outer-string graphs and some subclasses.

The class of outer-string graphs includes the circle graphs (i.e., graphs of intersections of chords of a circle), so any decision problem that is NP-hard for circle-graphs is also NP-hard for outer-string graphs. This includes, among others, the Coloring problem and the Hamiltonian Cycle problem [11, 5]. However, it does not include the maximum independent set problem, i.e., the problem where we are given a graph with vertex-weights (not necessarily uniform), and we want to find the maximum-weight vertex-set I such that no two vertices in I are adjacent.

The work in the current paper was inspired by a result from 2015 in which Keil, Mitchell, Pradhan and Vatshelle presented a poly-time algorithm for maximum independent set in outer-string graphs [14]. They assume that an outer-string representation R is given, and "poly-time" means polynomial in the size of R (typically measured by assuming that R uses only polygonal lines and counting the number of segments). Their algorithm runs in time $O(N^3)$ where N is the size of R.

Since the algorithm of Keil et al. [14] requires the representation to be given, the following question remains open: Given an outer-string graph G (but no outer-string representation), can we find a maximum independent set of G in polynomial time? One natural approach to this would be to try to find an outer-string representation of G. There are two obstacles here though. First, no algorithm is known to find such a representation (but this problem is also not known to be NP-hard). Second, even if such an algorithm were known, what would be the size N of the resulting outer-string representation? There are string graphs for which any string representation requires exponential size [16]. What can be said about the size of a representation required for outer-string graphs? This is the main topic of this paper.

1.1 Related results

We provide here an overview of some algorithmic results on string graphs. Since planar graphs are string graphs [7], all problems that are NP-hard for planar graphs remain NP-hard for string graphs. The converse statement is not true because there are problems that are polynomial for planar graphs (e.g. maximum clique) but NP-hard for string graphs [19].

String representations have been used to obtain better approximation algorithms, especially for independent set. Matoušek [18] showed that every string graph with m edges admits a vertex separator (a set S such that all components of $G - S$ have at most $\frac{2}{3}n$ vertices) of size $O(\sqrt{m}\log m)$. Fox and Pach conjectured that every string graph has a separator of size $O(\sqrt{m})$ [9]. This was proved, first for *k-intersecting string graphs* (any two strings intersect at most k times) [8] and very recently for all string graphs [17]. One example of a result based on separators is an n^{ε}-approximation algorithm for maximum independent set in k-intersecting string graphs by Fox and Pach [10]. Har-Peled and Quanrund [13] showed that separator theorems are applicable for approximation algorithms for all sparse string graphs. However, none of these results seems to lead to approximation algorithms with factors better than $O(n^{\varepsilon})$ for all string graphs.

A *segment graph* is a string graph that has a string representation in which all strings are line segments. Agarwal and Mustafa [1] proved that if all these segments intersect a given line, then an independent set of size $\sqrt{\alpha}$ can be computed in $O(n^3)$ time where α is the size of a maximum independent set. They also showed that any segment graph can be split into $O(\log n)$ subgraphs that are segment graphs for which all segments intersect one line. They used this to obtain an independent set of size $\sqrt{\alpha/\log(n/\alpha)}$ for all segment graphs.

1.2 Our contribution

In this paper we show that for some n-vertex outer-string graphs any outer-string representation requires $\Omega(2^{n/10})$ crossings, and consequently exponentially many segments. This result implies that the independent set algorithm of Keil et al. [14] does not run in polynomial time for all outer-string graphs, but only for those that have a polynomial-size outer-string representation. Our result also motivates exploration of algorithms whose running times have lower dependency on the size of representation.

We next explore graph classes that do have small string representations. We consider a natural subclass of string graphs, the *monotone string graphs*, where every string is a y-monotone curve, and argue that any such representation can be transformed into one of polynomial size. Combining this with Keil et al.'s algorithm implies that for monotone outer-string graphs, the maximum independent set is polynomial in n, however, the running time is rather large. We also study a special case where every monotone string has one endpoint on an enclosing strip. For this case we present a dynamic programming algorithm that finds a maximum independent set in $O(n^6)$ time and a 2-approximation in $O(n^3)$ time.

1.3 Outer-string graphs and apices

For our proof that some outer-string graphs require large representations, it will help to have a characterization of outer-string graphs. Although this characterization is simple (a similar approach has been used by Middendorf and Pfeiffer to characterize so-called cylinder graphs [20]), to our knowledge it has not been given before. Let G be a graph. The *apex graph* H of G is the graph obtained from G by adding a new vertex a connected to all vertices in G. The *subdivided apex graph* of G, denoted by G^+, is obtained from H by subdividing every edge incident to a. See Figure 1. One can easily show the following (see [6] for details):

▶ **Lemma 1.** *Graph G is an outer-string graph if and only if its subdivided apex graph G^+ is a string graph. Furthermore, any outer-string representation R of G can be turned into a string representation of G^+ without changing any curve of R.*

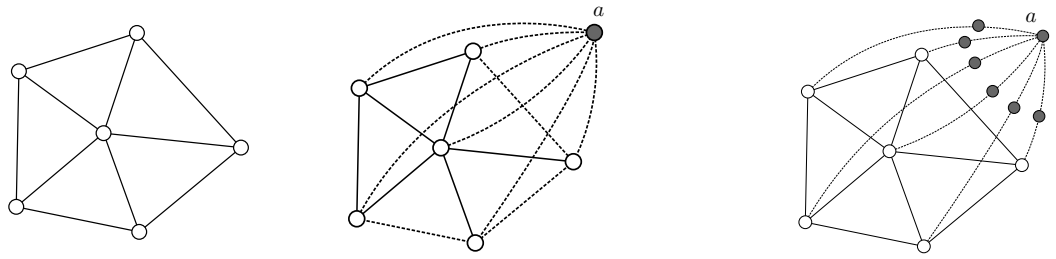


Figure 1 A graph G, its apex graph, and the subdivided apex graph G^+.

We have two corollaries from this that should be interesting in their own right. First, it is known that every string graph G with m edges has a string representation with $2^{O(m)}$ crossings per string. (This holds since string representations of G correspond to so-called weak realizations of another graph H with $O(m)$ edges [22], and weak realizations can be assumed to have at most 2^m crossings per string [23, 21].) Therefore, if G is outer-string, then the subdivided apex-graph of G has a string representation with $2^{O(m)}$ crossings per string. Deleting the added vertices, we get the following corollary.

▶ **Corollary 2.** *If G is an outer-string graph with m edges, then it has an outer-string representation with $2^{O(m)}$ crossings per string.*

Secondly, while it was long known that string graph recognition is NP-hard [15], proving that it is in NP was a long-standing open question until proved by Schaefer et al. [22]. For any graph G we can construct the subdivided apex graph G^+ in polynomial time. Combining this with Lemma 1 implies the following non-trivial result.

▶ **Corollary 3.** *The problem of recognizing outer-string graphs is in NP.*

Naturally one wonders whether recognizing outer-string graphs is also NP-hard. This problem is open.

2 Exponential-sized Outer-string Representations

Now we construct a graph that requires exponentially many intersections in any string representation. This re-proves a result of Kratochvíl and Matoušek [16], but our graph is different (although inspired by their construction), and can be used to prove the same for outer-string representations later.

For any integer $k \geq 1$ construct graph G_k as follows. For $0 \leq i \leq k$ we add two vertices x_i, y_i, and an edge (x_i, y_j) for every $j \geq i$. We surround this graph with a gadget that forces these vertices to appear in a certain order. This was done in [16] with a grid-like structure, but we use a cycle C instead, in the same way that Cardinal et al. [3] used a cycle to enforce order for their representations.[4] Specifically, let $C = c_0, c_1, \ldots, c_{K-1}$ be a cycle with $K := 8k + 8$ vertices. We connect every 4th vertex of C to one of the vertices $\{x_i, y_i\}_{i=0,\ldots,n}$, in order (along the cycle) $x_0, x_1, y_0, x_2, y_1, x_3, y_2, \ldots, x_i, y_{i-1}, x_{i+1}, y_i, \ldots, x_k, y_{k-1}, y_k$. Let $\ell(x_i)$ [resp. $\ell(y_i)$] be the index of the vertex of C that is adjacent to x_i [resp. y_i], thus $\ell(x_0) = 0$, $\ell(x_1) = 4$, etc. See also Figure 2. This finishes the construction of G_k. As before, let G_k^+ be the subdivided apex graph of G_k with apex vertex a. We use s_j for the subdivision-vertex incident to vertex $c_j \in C$.

Figure 3 illustrates an outer-string representation of G_k, which can be converted into a string representation of its subdivided apex graph G_k^+ (see Lemma 1). Note that $\mathbf{y_k}$ and $\mathbf{x_0}$ intersect 2^{k-1} times. We now argue that this is required.

▶ **Lemma 4.** *In any string representation of G_k^+, curve $\mathbf{y_k}$ intersects curve $\mathbf{x_0}$ at least 2^{k-1} times.*

Proof. Fix a string representation R_k^+ of G_k^+. Delete from it all strings of all subdivision vertices s_{2i+1} for $0 \leq i < K/2$ (these won't be needed). Also, we know that s_{2i} has only two neighbours (a and c_{2i}), and we can hence shorten its string such that $\mathbf{s_{2i}}$ has exactly two intersections, one with $\mathbf{a}$ and one with $\mathbf{c_{2i}}$ [15]. Likewise $\mathbf{c_{2i+1}}$ intersects only two other strings ($\mathbf{c_{2i}}$ and $\mathbf{c_{2i+2}}$) since we deleted $\mathbf{s_{2i+1}}$, and we may hence shorten it such that $|\mathbf{c_{2i}} \cap \mathbf{c_{2i+1}}| = 1 = |\mathbf{c_{2i+1}} \cap \mathbf{c_{2i+2}}|$.

So for any $0 \leq j < K$, we have a unique point in $\mathbf{c_j} \cap \mathbf{c_{j+1}}$ (addition for all vertices in C is mod K). We use $\mathbf{c_j}[c_{j-1}, c_{j+1}]$ to denote the (unique) stretch of $\mathbf{c_j}$ between $\mathbf{c_{j-1}} \cap \mathbf{c_j}$ and $\mathbf{c_j} \cap \mathbf{c_{j+1}}$. Crucial for our argument will be a curve defined by following the strings of cycle C: define $\mathbf{C}$ to be $\bigcup_{j=0}^{K-1} \mathbf{c_j}[c_{j-1}, c_{j+1}]$ and observe that it is a closed simple curve in the plane, hence splits the plane into the inside and outside. We now make a sequence of observations:

- Since the apex-vertex is not adjacent to any vertex of C, curve $\mathbf{a}$ is disjoint from $\mathbf{C}$ and hence resides inside or outside. By symmetry, we may assume that $\mathbf{a}$ is outside $\mathbf{C}$.

[4] The correctness for their gadget was only argued for outer-1-string representations, and so we cannot use it as a black box, but the idea is the same.

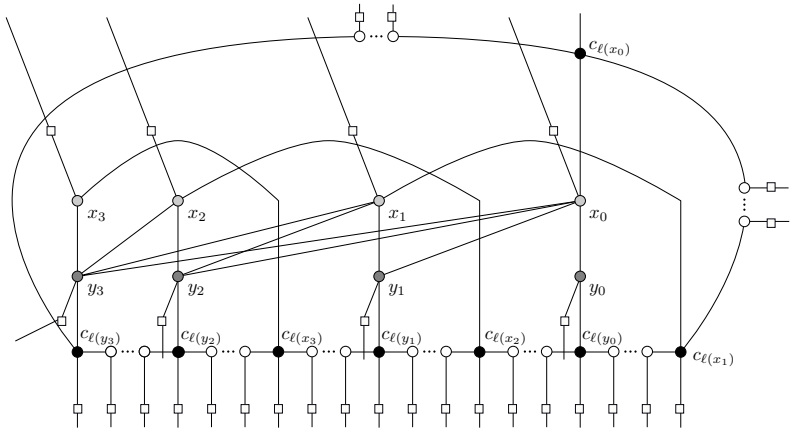


Figure 2 The graph G_3^+. The apex vertex a is not shown. Subdivision vertices are squares.

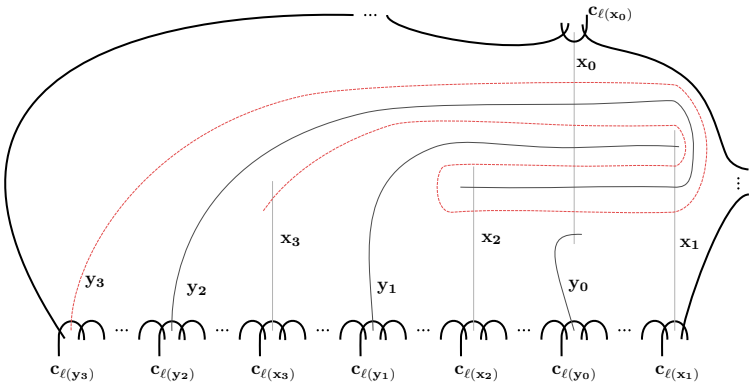
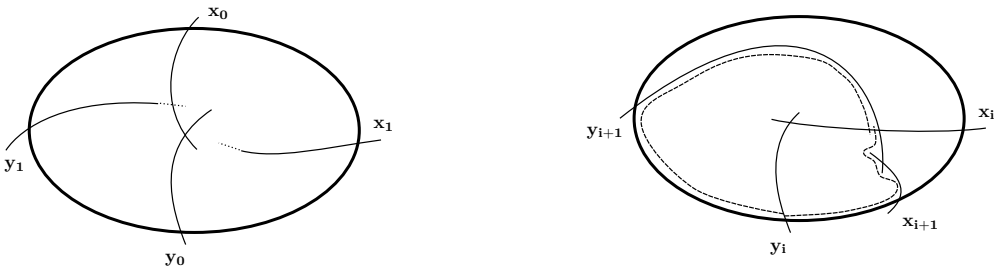


Figure 3 An outer-string representation of G_3. String $\mathbf{y_3}$ is red (dashed) for ease of legibility.



(a) The base case.

(b) Two possible routes for $\mathbf{y_i}$.

Figure 4 In the base case, $\mathbf{y_1}$ must cross $\mathbf{x_0}$. In the induction step, a route for $\mathbf{y_{i+1}}$ gives two possible routes for $\mathbf{y_i}$ to $\mathbf{x_i}$.

- For any $0 \leq i \leq k$, curve $\mathbf{x_i}$ has a point outside $\mathbf{C}$. Namely, there exists a subdivision-vertex s_{x_i} with unique neighbors a and x_i. Since neither a nor s_{x_i} have a neighbor on $\mathbf{C}$, and $\mathbf{a}$ is outside $\mathbf{C}$, therefore so is $\mathbf{s_{x_i}}$, and so any point in $\mathbf{s_{x_i}} \cap \mathbf{x_i}$ is outside $\mathbf{C}$.
- For any $0 \leq i \leq k$, curve $\mathbf{x_i}$ has a point inside $\mathbf{C}$. Specifically, for any $j \geq i$ any point in $\mathbf{x_i} \cap \mathbf{y_j}$ (which exists since there is an edge (x_i, y_j)) must be inside $\mathbf{C}$ for any $j > i$. For otherwise we could use a point in $\mathbf{x_i} \cap \mathbf{y_j}$ outside $\mathbf{C}$ to find a drawing of K_4 with all vertices on one face, an impossibility. (Details are in [6].)
- Thus for any $0 \leq i \leq k$, curve $\mathbf{x_i}$ has points both inside and outside $\mathbf{C}$. So $\mathbf{x_i}$ must intersect $\mathbf{C}$, which is possible only at $\mathbf{c}_{\ell(\mathbf{x_i})}$.
- Similarly $\mathbf{y_j}$ intersects at a point on $\mathbf{C}$ for all $0 \leq j \leq k$, and this intersection must happen on $\mathbf{c}_{\ell(\mathbf{y_j})}$.

We are now almost ready to argue the number of intersections of $\mathbf{y_k}$ with $\mathbf{x_0}$, which will happen by induction on k. However, to argue the induction step it helps to permit that some curves do not intersect. We hence use the following type of representation:

▶ **Definition 5.** A *weak outer G_i-representation* is a collection R'_i of curves $\mathbf{C}, \mathbf{x_0}, \mathbf{y_0}, \ldots, \mathbf{x_i}, \mathbf{y_i}$ that satisfies the following:

1. $\mathbf{C}$ is a simple closed curve such that all other curves of R'_i are on or inside $\mathbf{C}$.
2. The curves $\mathbf{x_j}$ and $\mathbf{y_j}$ (for $0 \leq j \leq i$) intersect each other.
3. Each of the curves $\mathbf{x_j}$ and $\mathbf{y_j}$ (for $0 \leq j \leq i$) intersects $\mathbf{C}$ exactly once.
4. These intersections with $\mathbf{C}$ occur in order $\mathbf{x_0}$, $\mathbf{x_1}$, $\mathbf{y_0}$, $\mathbf{x_2}$, $\mathbf{y_1}$, $\mathbf{x_3}$, $\mathbf{y_2}, \ldots, \mathbf{x_i}$, $\mathbf{y_{i-1}}$, $\mathbf{y_i}$.
5. The curves $\mathbf{x_r}$ and $\mathbf{y_j}$ (for $0 \leq r < j \leq i$) may or may not intersect each other.
6. No other curves are allowed to intersect each other.

It is easy to see ([6] has details) that for any $0 \leq i \leq k$ we can find a weak outer G_i-representation for which all curves reside within the corresponding curves of R_k^+. The theorem hence holds once we have shown the following:

▶ Claim 1. In any weak outer G_i-representation, curve $\mathbf{y_i}$ intersects $\mathbf{x_0}$ at least 2^{i-1} times.

We proceed by induction on i. Consider the base case $i = 1$ (see Figure 4(a); for legibility we extend curves slightly beyond $\mathbf{C}$). The order in which curves intersect $\mathbf{C}$ is $\mathbf{x_0}, \mathbf{x_1}, \mathbf{y_0}, \mathbf{y_1}$, and the combined curve $\mathbf{x_0} \cup \mathbf{y_0}$ splits $\mathbf{C}$ into two parts. Curves $\mathbf{x_1}$ and $\mathbf{y_1}$ intersect $\mathbf{C}$ in different parts. To create an intersection point $\mathbf{x_1} \cap \mathbf{y_1}$, one of them must cross paths $\mathbf{y_0} \cup \mathbf{x_0}$. Such a crossing must be between $\mathbf{y_1}$ and $\mathbf{x_0}$ (no other crossings are allowed). So, $\mathbf{y_1}$ intersects $\mathbf{x_0}$ at least once.

Assume now that the claim holds for some i, and study a weak outer G_{i+1}-representation. Curve $\mathbf{y_{i+1}}$ is separated from curve $\mathbf{x_{i+1}}$ by $\mathbf{x_i} \cup \mathbf{y_i}$. Thus, curve $\mathbf{y_{i+1}}$ has to intersect $\mathbf{x_i}$ on its way to $\mathbf{x_{i+1}}$. On the way to $\mathbf{x_i}$, it has to create at least 2^{i-1} intersections with $\mathbf{x_0}$, otherwise we could re-route $\mathbf{y_i}$ and use fewer crossings between $\mathbf{y_i}$ and $\mathbf{x_0}$. More precisely (refer to Figure 4(b)), $\mathbf{y_i}$ could be re-routed to stay in the proximity of the cycle $\mathbf{C}$ until it reaches $\mathbf{y_{i+1}} \cap \mathbf{C}$ and then follow $\mathbf{y_{i+1}}$ until reaching $\mathbf{x_i}$. Along this new route (following $\mathbf{y_{i+1}}$) curve $\mathbf{y_i}$ might intersect neighbors of y_{i+1}, but all those neighbors are allowed to be neighbors of y_i as well, so this is (after deleting $\mathbf{y_{i+1}}$ and $\mathbf{x_{i+1}}$) a weak outer G_i-representation with less than 2^{i-1} points in $\mathbf{y_i} \cap \mathbf{x_0}$. This contradicts the induction hypothesis. So, $\mathbf{y_{i+1}}$ intersects $\mathbf{x_0}$ at least 2^{i-1} times on the way from $\mathbf{C} \cap \mathbf{y_{i+1}}$ to $\mathbf{y_{i+1}} \cap \mathbf{x_i}$.

On the way from $\mathbf{x_i}$ to $\mathbf{x_{i+1}}$, curve $\mathbf{y_{i+1}}$ needs to create another 2^{i-1} crossings with $\mathbf{x_0}$, otherwise we could re-route $\mathbf{y_i}$ and use fewer crossings as follows: $\mathbf{y_i}$ stays in the proximity of the cycle curves until it reaches $\mathbf{x_{i+1}} \cap \mathbf{C}$ and then follows $\mathbf{x_{i+1}}$ and $\mathbf{y_{i+1}}$. Thus $\mathbf{y_{i+1}}$ crosses $\mathbf{x_0}$ at least 2^i times as desired. ◀

In consequence, we have:

▶ **Theorem 6.** *For any $k \geq 1$, there exists a graph G_k with $O(k)$ vertices that has an outer-string representation, but any outer-string representation of G_k requires two strings to intersect at least 2^{k-1} times.*

Proof. We use graph G_k defined earlier; it has $10k + 10$ vertices total. By Lemma 4, any string representation of G_k^+ requires at least 2^{k-1} intersections between y_k and x_0. Since any such representation can be obtained from an outer-string representation of G_k without changing any string of G_k (see Lemma 1), any outer-string representation of G_k requires at least 2^{k-1} intersections between $\mathbf{y_k}$ and $\mathbf{x_0}$. ◀

Since line segments intersect at most once, any polygonal outer-string representation of G_k hence must have a string with at least $\sqrt{2^{k-1}} = 2^{(k-1)/2} \in 2^{\Omega(n)}$ segments.

3 Monotone string representations

In the previous section, we showed that outer-string graphs sometimes require an exponential number of segments in any outer-string representation. Naturally, one wonders whether there are any natural subclasses of string graphs that have polynomial-size representations.

In this section, we prove that there are string representations of polynomial size if the graph is a *monotone string graph*. By this we mean that it has a string representation where every curve is y-monotone, i.e., intersects any horizontal line at most once. Monotone string graphs have been studied before (e.g. in the context of coloring [25]), but to our knowledge the following is new:

▶ **Theorem 7.** *Let G be an n-vertex m-edge graph with a monotone string-representation R. Then G has a monotone string-representation R' with at most $2n(n+m)$ segments.*

Proof. We may assume that no two y-coordinates of crossings or endpoints in R coincide, and no string has its endpoint on another string. Define a *layer-set* $\mathcal{Y}$ of y-coordinates as follows: (1) For every vertex v, add to $\mathcal{Y}$ the y-coordinates of the bottom and top endpoints of $\mathbf{v}$. (2) For every edge $e = (v, w)$, pick one point p in $\mathbf{v} \cap \mathbf{w}$ and add to $\mathcal{Y}$ the y-coordinates $y_e^- := y(p) - \varepsilon$ and $y_e^+ := y(p) + \varepsilon$, where ε is small enough such that no other intersections or endpoints of curves happen within this range. See also Figure 5. Now create R' by defining, for each vertex v, the curve $\mathbf{v'}$ as a poly-line that connects, from bottom to top, the points

where $\mathbf{v}$ intersects a horizontal line with y-coordinate in $\mathcal{Y}$. In the rest of the proof we verify that this represents the same graph and satisfies all conditions.

For any $y \in \mathcal{Y}$, define ℓ_Y to be the horizontal line with y-coordinate Y; we call ℓ_Y a *layer*. To define the new curve $\mathbf{v}'$ for a vertex v, let $y_1 < \cdots < y_d$ be all those values $y_i \in \mathcal{Y}$ for which ℓ_{y_i} intersects $\mathbf{v}$. Now let $\mathbf{v}'$ be the poly-line $\mathbf{v} \cap \ell_{y_1}, \mathbf{v} \cap \ell_{y_2}, \ldots, \mathbf{v} \cap \ell_{y_d}$. (These intersection points are unique since $\mathbf{v}$ is monotone.) Curve $\mathbf{v}'$ is monotone and has at most $2m + 2n - 1$ segments.

It remains to argue that R' represents the same graph as R did. If $e = (v, w)$ is an edge, then $\mathbf{v}'$ crosses $\mathbf{w}'$ between the two layers that were added just above and below a point in $\mathbf{v} \cap \mathbf{w}$.

For the other direction, let us assume that curves $\mathbf{v}'$ and $\mathbf{w}'$ cross in R', say at point c. The crossing c cannot lie on a layer ℓ, because both $\mathbf{v}'$ and $\mathbf{w}'$ cross ℓ at the same points as $\mathbf{v}$ and $\mathbf{w}$ did, and $\mathcal{Y}$ was chosen so that no layer contains crossings of R.

So c lies between two consecutive layers, say ℓ and ℓ'. After possible renaming, assume that $\mathbf{v}' \cap \ell$ lies to the left of $\mathbf{w}' \cap \ell$. Since the curves use line segments between layers and there is a crossing, we must have the reverse order on ℓ', i.e., $\mathbf{v}' \cap \ell'$ lies to the right of $\mathbf{w}' \cap \ell'$.

But recall that we chose $\mathbf{v}'$ such that $\mathbf{v}' \cap \ell = \mathbf{v} \cap \ell$, and similarly for w and ℓ'. Therefore, in R we also had $\mathbf{v}$ to the left of $\mathbf{w}$ on ℓ and to the right of $\mathbf{w}$ on ℓ'. Curves $\mathbf{v}$ and $\mathbf{w}$ are y-monotone in the stretch between ℓ and ℓ'. It follows that the two curves $\mathbf{v}$ and $\mathbf{w}$ cross somewhere within this stretch. Therefore (v, w) is an edge of the graph as required. ◀

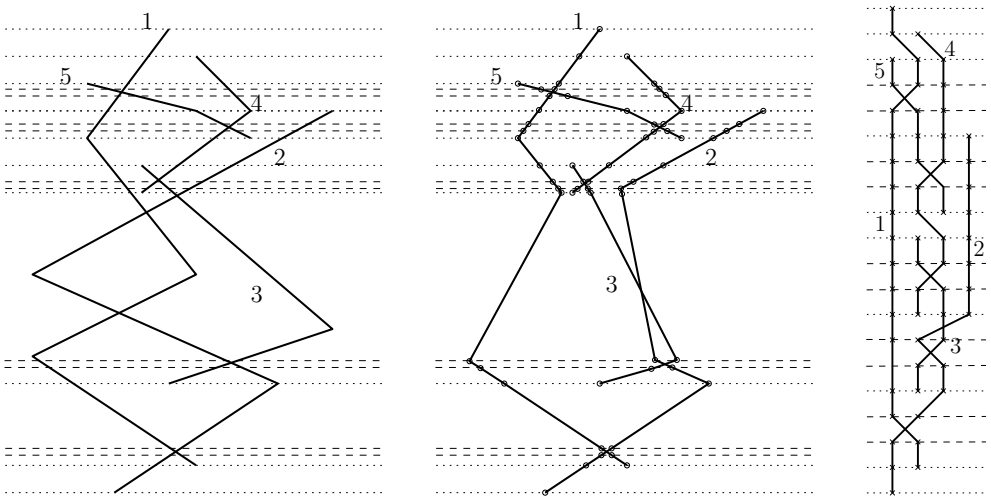


Figure 5 A monotone string-representation of C_5, an application of our algorithm, and re-assigning coordinates to obtain an $n \times (2m + 2n)$-grid.

We note that R' can be assumed to reside on an $n \times (2m + n)$-grid. Namely, each curve consists of line segments that connect consecutive layers. We can re-assign y-coordinates in $\{1, \ldots, 2m + 2n\}$ to the layers, and re-assign x-coordinates in $\{1, \ldots, n\}$ to the points where curves intersect layers, and the same line segments will cross between consecutive layers, hence we obtain a string representation of the same graph. See Figure 5.

One drawback of our proof is that it needs an explicit representation R to create the polynomial-sized representation R'. It remains open how to find such a representation R', given just the graph.

4 Independent set in monotone outer-string graphs

Keil et al. presented an algorithm for (weighted) independent set on outer-string graphs that runs in time $O(N^3)$ (as before, N is the size of an outer-string representation) [14]. However, due to Theorem 6, N may need to be in $2^{\Omega(n)}$. In this section we study the independent set problem on monotone string graphs, which have a polynomial-size representation by Theorem 7. Since planar graphs are segment graphs [4] (hence monotone string graphs), and since maximum independent set is NP-hard for planar graphs [12], we have:

▶ Proposition 1. Maximum independent set is NP-hard even for monotone string graphs.

We therefore turn our attention to monotone outer-string graphs. Here, we know from Keil et al.'s result that the maximum independent set problem is solvable in polynomial time in the size of representation, and from Theorem 7 that there exists a representation with size $N \in O(nm)$ and at most $O(m+n)$ line segments per string. Presuming such a representation is given, we can hence solve the independent set problem in $O(n^3m^3)$ time. We now show that for two special cases of monotone outer-string graphs, a better run-time can be achieved.

▶ **Definition 8.** Let G be a monotone outer-string graph. We say that G is *strip-grounded* if there exists a monotone string representation of G with a bounding rectangle ρ such that all strings have one end at the top or bottom side of ρ. We say that G is *line-grounded* if all strings have one end on the bottom side of ρ.

Figures 6 and 7 illustrate line-grounded and strip-grounded graphs, respectively. We may, after shortening some strings, assume that no string in such a representation touches both the bottom and the top of ρ. For a string $\mathbf{v}$, we use $b(v)$ and $t(v)$ to denote the y-coordinates of the bottom and top endpoints of $\mathbf{v}$, respectively. For ease of description, we add two negative-weight dummy vertices with strings along the left and right sides of ρ (no optimal solution will include these two vertices/strings). Enumerate the *bottom-grounded vertices* (i.e., vertices whose strings attach at the bottom side of ρ) as $v_1, \ldots, v_b$, from left to right by bottom endpoint. Enumerate the *top-ground vertices* as $u_1, \ldots, u_t$, from left to right by top endpoint. Here, $v_1 = u_1$ and $v_b = u_t$ are the dummy vertices.

4.1 Line-grounded monotone string graphs

We first show how to find the maximum independent set in a line-grounded monotone string graph G; this will be a useful subroutine later. We only have vertices $v_1, \ldots, v_b$ (with $b = n+2$ due to the dummy vertices). We proceed by dynamic programming, and define sub-problems as follows (a similar technique has been used in [1] for computing approximate maximum independent set of segments that cross a straight line). For any pair (i,j), with $1 \le i < j \le b$ and $(v_i, v_j) \notin E$, define $S(i,j)$ to be the set of vertices $v_k \in \{v_{i+1}, \ldots, v_{j-1}\}$ that satisfy $t(v_k) \le \min\{t(v_i), t(v_j)\}$, $(v_k, v_i) \notin E$ and $(v_k, v_j) \notin E$. Put differently, $S(i,j)$ contains every vertex v_k for which $\mathbf{v_k}$ is strictly within the region bounded by $\mathbf{v_i}$, the bottom side of ρ, $\mathbf{v_j}$, and the horizontal line with y-coordinate $\min\{t(v_i), t(v_j)\}$ (see Figure 6). Due to the dummy vertices, we have $S(1,b) = V$.

Let $w(v)$ be the weight of vertex v, and set $W(i,j)$ to be the weight of a maximum independent set in $S(i,j)$.

▶ Claim 2. $W(i,j) = \begin{cases} 0 & \text{if } S(i,j) \text{ is empty} \\ \max_{v_k \in S(i,j)} W(i,k) + W(k,j) + w(v_k) & \text{otherwise.} \end{cases}$

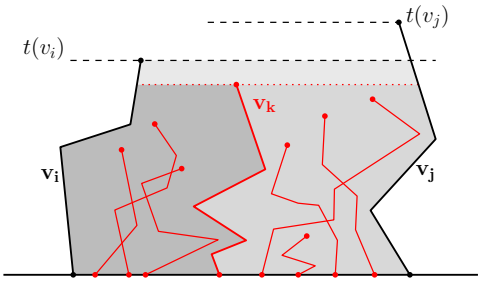


Figure 6 Line-grounded strings, and an illustration of the formula for $W(i,j)$.

Proof. See Figure 6 for an illustration of this proof. Consider an optimal solution I^* for $S(i,j)$. Let v_k be the vertex that maximizes $t(v_k)$ among the vertices in I^* (if there is no such v_k then $S(i,j) = \emptyset$ and the equality holds). Let v be some other vertex in I^*. Since I^* is an independent set, $\mathbf{v}$ does not intersect $\mathbf{v_k}$. It also intersects neither $\mathbf{v_i}$ nor $\mathbf{v_j}$ by definition of $S(i,j)$. Finally $t(v) \leq t(v_k)$ by choice of v_k. It follows that $v \in S(i,k)$ or $v \in S(k,j)$. So $I^* - \{v_k\}$ induces two independent sets for $S(i,k)$ and $S(k,j)$. So "$\leq$" holds for this choice of v_k, and even more so for the maximum among all v_k in $S(i,j)$.

For the other direction, let k be the index where the maximum is achieved and fix maximum independent sets I_i and I_j of $S(i,k)$ and $S(k,j)$. Observe that no string of I_i can intersect one in I_j since they reside within disjoint regions, and neither of them can intersect $\mathbf{v_k}$ by definition of $S(i,k)$ and $S(k,j)$. So $I_i \cup I_j \cup \{v_k\}$ is an independent set of $S(i,j)$ and "$\geq$" holds. ◀

By computing $S(1,b)$ recursively with standard dynamic programming techniques, we can hence find the maximum independent set of G. We briefly discuss the run time. To find set $S(i,j)$, we mark all neighbours of v_i, all neighbours of v_j, and all vertices v with $t(v) > \min\{t(v_i), t(v_j)\}$. Then we take all unmarked vertices in $\{v_{i+1}, \ldots, v_{j-1}\}$; clearly this takes $O(n)$ time per set $S(i,j)$. Evaluating the recursive formula takes $O(n)$ time as well, and since we have $O(n^2)$ subproblems, the overall run-time is $O(n^3)$. (Note that for this algorithm, we do not even need an explicit line-grounded monotone string representation: it suffices to have graph G, and the coordinates of the top and bottom endpoints, together with the promise that they correspond to such a representation.)

▶ **Theorem 9.** *Given a vertex-weighted graph G with a line-grounded monotone string representation, we can compute the maximum-weight independent set of G in $O(n^3)$ time.*

4.2 Strip-grounded monotone string graphs

Now we turn to strip-grounded monotone string graphs. First note that by applying the algorithm for line-grounded monotone string graphs twice (once for the bottom-grounded vertices and once for the top-grounded vertices), we immediately obtain a 2-approximation algorithm, which runs in $O(n^3)$ time. At the price of an increased run-time, we show how to solve this problem optimally. For this, we need a more complicated set of subproblems:

Let $\mathbf{v}$ be a bottom-grounded string and $\mathbf{u}$ be a top-grounded string such that $(v,u) \notin E$ and $t(v) \geq b(u)$. We say that a vertex x lies *between v and u* if there exists a horizontal line ℓ that intersects all of $\mathbf{v}, \mathbf{u}, \mathbf{x}$, and for which the point $\ell \cap \mathbf{x}$ lies between the points $\ell \cap \mathbf{v}$ and $\ell \cap \mathbf{u}$. Define the following sets (see also Figure 7):

- Let $1 \leq i \leq j \leq b$ and $1 \leq \alpha \leq \beta \leq t$ be indices such that $\{v_i, u_\alpha, v_j, u_\beta\}$ is an independent set, and further $t(v_i) \geq b(u_\alpha)$ and $t(v_j) \geq b(u_\beta)$. Define $S(i,\alpha;j,\beta)$ to be

all those vertices in $\{v_{i+1}, \ldots, v_{j-1}\} \cup \{u_{\alpha+1}, \ldots, u_{\beta-1}\}$ that are adjacent to none of $v_i, v_j, u_\alpha, u_\beta$, and do not lie between v_i and u_α or between v_j and u_β.

- Let $1 \le i \le k \le b$ and $1 \le \alpha \le t$ be indices such that $\{v_i, u_\alpha, v_k\}$ is an independent set and $t(v_i) \ge b(u_\alpha)$. Define $S^{SW}(i, \alpha; k)$ to be all those vertices v in $\{v_{i+1}, \ldots, v_{k-1}\}$ that are adjacent to none of v_i, v_k, u_α, do not lie between v_i and u_α, and for which $t(v) \le t(v_k)$.
- We symmetrically define $S^{SE}(k; j, \beta)$, $S^{NW}(i, \alpha; \gamma)$ and $S^{SW}(\gamma; j, \beta)$. See also Figure 7.
- Finally we also need the set $S(i, j)$ defined earlier (we denote it $S_v(i, j)$ since it uses the bottom-grounded vertices), and symmetrically set $S_u(\alpha, \beta)$ for top-grounded vertices.

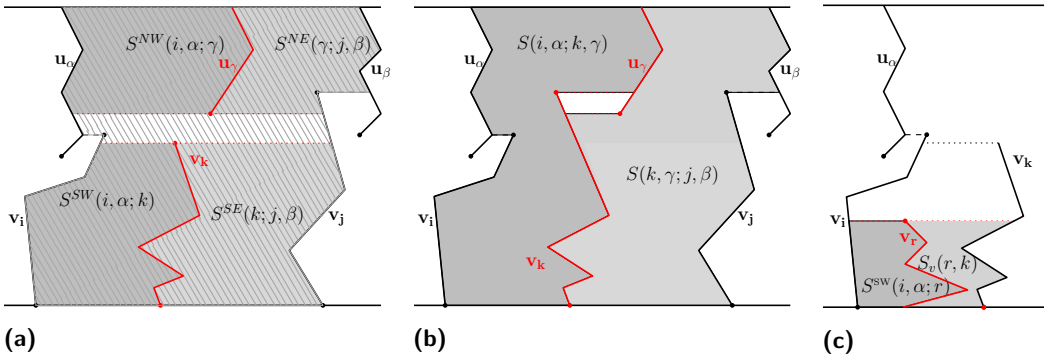


Figure 7 A strip-grounded graph. Strings in $S(i, \alpha; j, \beta)$ must be in the striped region. We illustrate recursive formulas (a) for $W(i, \alpha, j, k)$ for $t(v_k) < b(u_\gamma)$, (b) for $W(i, \alpha, j, k)$ for $t(v_k) \ge b(u_\gamma)$, and (c) for $W^{SW}(i, \alpha; k)$.

Let $W(i, j; \alpha, \beta)$ be the weight of a maximum independent set in subgraph induced by vertex set $S(i, \alpha; j, \beta)$, and similarly for all other sets. We already had the formula for $W_v(i, j)$ (Claim 2), and a symmetric one holds for $W_u(\alpha, \beta)$. With much the same proof one can show (see also Figure 7(c)):

▶ Claim 3. $W^{SW}(i, k; \alpha) = 0$ if $S^{SW}(i, k; \alpha)$ is empty. Otherwise,

$$W^{SW}(i, k; \alpha) = \max_{v_r \in S^{SW}(i,k;\alpha)} W^{SW}(i, r; \alpha) + W_v(r, k) + w(v_k).$$

The formula for W^{SW}, W^{NE} and W^{NW} are symmetric. As for $W(i, j; \alpha, \beta)$, based on whether the maximum independent set contains bottom-grounded or top-grounded vertices, and how they interact, one can show the following formula:

▶ Claim 4. $W(i, j; \alpha, \beta) = 0$ if $S(i, \alpha; j, \beta)$ is empty. Otherwise, it is the maximum of

$$\begin{aligned}
&\max_{v_k \in S(i,\alpha;j,\beta)} && W^{SW}(i, \alpha; k) + W^{SE}(k; j, \beta) + w(v_k),\\
&\max_{u_\gamma \in S(i,\alpha;j,\beta)} && W^{NW}(i, \alpha; \gamma) + W^{NE}(\gamma; j, \beta) + w(u_\gamma),\\
&\max_{v_k, u_\gamma \in S(i,\alpha;j,\beta), t(v_k) < b(u_\gamma)} && W^{SW}(i, \alpha; k) + W^{SE}(k; j, \beta) + w(v_k)\\
& && +W^{NW}(i, \alpha; \gamma) + W^{NE}(\gamma; j, \beta) + w(u_\gamma), \text{ and}\\
&\max_{v_k, u_\gamma \in S(i,\alpha;j,\beta), t(v_k) > b(u_\gamma), (v_k, u_\gamma) \notin E} && W(i, k; \alpha, \gamma) + W(k, j; \gamma, \beta) + w(v_k) + w(u_\gamma)
\end{aligned}$$

Proof. To show '$\ge$', observe that each term of the maximum on the right-hand side corresponds to two or four independent sets in two or four regions defined by the parameters. As one easily verifies, these regions are disjoint for all cases, and none of them contains $\mathbf{v_k}$ and/or $\mathbf{u_\gamma}$. We can hence combine these independent sets and add v_k and/or u_γ, and obtain an independent set for $S(i, \alpha; j, \beta)$. The optimum independent set cannot be smaller.

To prove '$\leq$', consider an optimal solution I^* for $S(i,\alpha;j,\beta)$. We may assume that I^* is non-empty; else $S(i,\alpha;j,\beta)$ is empty and the equation holds. We distinguish cases:

Case 1: I^* contains no top-grounded vertex. Since I^* is non-empty, it therefore contains some v_k with $i < k < j$. Let v_k be the vertex that maximizes $t(v_k)$. With this choice, any other vertex in I^* is bottom-grounded and belongs to $S^{SW}(i,\alpha;k)$ or $S^{SE}(k;j,\beta)$, depending on whether its index is before or after k. Therefore $I^* - \{v_k\}$ splits into two independent sets for $S^{SW}(i,\alpha;k)$ and $S^{SE}(k;j,\beta)$, and $w(I^*) + w(v_k) \leq W^{SW}(i,\alpha;k) + W^{SE}(k;j,\beta)$.

Case 2: I^* contains no bottom-grounded vertex. Symmetrically then one shows that $w(I^*) \leq W^{NW}(i,\alpha;\gamma) + W^{NE}(\gamma;j,\beta) + w(u_\gamma)$ where u_γ is the top-grounded vertex in I^* that minimizes $b(u_\gamma)$.

Case 3: I^* contains a bottom-grounded vertex v_k and a top-grounded vertex u_γ, but for any two such vertices we have $t(v_k) < b(u_\gamma)$. Choose v_k so that it maximizes $t(v_k)$ and u_γ so that it minimizes $b(u_\gamma)$. Then any other bottom-grounded vertex v in I^* satisfies $t(v) \leq t(v_k)$ and so belongs to $S^{SW}(i,\alpha;k)$ or $S^{SE}(k;j,\beta)$. Any other top-grounded vertex u in I^* satisfies $b(u) \geq b(u_\gamma)$ and so belongs to $S^{NE}(i,\alpha;\gamma)$ or $S^{NE}(\gamma;j,\beta)$. Thus $I^* - \{v_k, u_\gamma\}$ splits into four independent sets for these four vertex sets, hence $w(I^*) \leq W^{SW}(i,\alpha;k) + W^{SE}(k;j,\beta) + w(v_k) + W^{NW}(i,\alpha;\gamma) + W^{NE}(\gamma;j,\beta) + w(u_\gamma)$.

Case 4: I^* contains a bottom-grounded vertex v_k and a top-grounded vertex u_γ with $t(v_k) \geq b(u_\gamma)$. Thus, any line ℓ with y-coordinate in $[b(u_\gamma), t(v_k)]$ intersects both $\mathbf{v_k}$ and $\mathbf{u_\gamma}$. We may assume that any such line ℓ intersects no other string of I^* in the range between $\ell \cap \mathbf{v_k}$ and $\ell \cap \mathbf{u_\gamma}$, else we can replace either v_k or u_γ with the intersected string. Thus no other vertex x in I^* is between v_k and u_γ. Therefore any vertex $x \neq v_k, u_\gamma$ in I^* belongs to either $S(i,\alpha;k,\gamma)$ or to $S(k,\gamma;j,\beta)$. So $I^* - \{v_k, u_\gamma\}$ splits into two independent sets for these two subsets. This proves that $w(I^*) \leq W(i,\alpha;k,\gamma) + W(k,\gamma;j,\beta) + w(v_k) + w(u_\gamma)$. ◀

Since $S(1,a;1,b) = V$ with our choice of dummy vertices, therefore we can compute the maximum independent set in G with dynamic programming. To analyze its run-time, observe that we have defined $O(n^4)$ sets. To compute each set, we need to test quickly whether x is between v_i and u_α for some independent set $\{x, v_i, u_\alpha\}$. We first test whether there exists some Y with $b(u_\alpha) \leq Y \leq t(v_i)$ and $b(x) \leq Y \leq t(x)$; otherwise x is surely not between them. If there is such a Y, then next find the points p_u, p_v, p_x where the horizontal line with y-coordinate Y intersects the three strings, and test their order. Recall that we assumed the strings to have $O(m+n)$ segments, so finding these points (and hence testing whether x is between v_i and u_α) can be done with binary search in $O(\log n)$ time.

With this, each set can be found in $O(n^2)$ time. For example, to find $S(i,\alpha;j,\beta)$, scan the vertices $v_{i+1}, \ldots, v_{j-1}$ and $u_{\alpha+1}, \ldots, u_{\beta-1}$. For each, test in $O(n)$ time whether it is non-adjacent to $v_i, u_\alpha, v_j, u_\beta$, and test in $O(\log n)$ time that it is neither between v_i and u_α, nor between v_j and u_β. The computation for the other types of sets is similar.

Given all the sets, the evaluation of the formula can be done in $O(n^2)$ time per set, or $O(n^6)$ total for all $O(n^4)$ sets. Therefore, we get the following theorem.

▶ **Theorem 10.** *The maximum independent set in a vertex-weighted graph with a strip-grounded monotone string representation can be computed in $O(n^6)$ time.*

5 Conclusions

In this paper, we studied graphs that do or do not have string representations of polynomial size. We argued that for some outer-string graphs any outer-string representation must have

exponential size. On the other hand, all monotone string graphs have a string representation of polynomial size. Inspired by an algorithm of Keil et al. for maximum independent set for outer-string graphs, we give an algorithm for maximum independent set for monotone strip-grounded outer-string graphs, whose run-time is $O(n^6)$, presuming we are given such a representation of polynomial size. We leave a number of open problems:

- We have introduced some variants of string graphs (e.g., monotone string graphs, monotone strip-grounded string graphs). What is the complexity of recognizing these graphs classes and finding corresponding representations? Note that it is not even known whether recognizing outer-string graphs is NP-hard (we proved that it is in NP).
- What is the complexity of recognizing whether a graph has a string representation (or outer-string representation) with at most k segments?
- Is there an algorithm for independent set on outer-string graphs that is polynomial in n? Our results show that this is not possible if we use an explicit description of a string representation. But perhaps the string representation could be given implicitly in a different way? Or perhaps it could be described (similarly as in [22]) with $O(\log N)$ bits, by listing how it intersects a (suitably) chosen triangulation? Note that $\log N \in O(m)$, so this would be polynomial.

References

1. P. K. Agarwal and N. H. Mustafa. Independent set of intersection graphs of convex objects in 2D. *Comput. Geom.*, 34(2):83–95, 2006.
2. S. Cabello and M. Jejčič. Refining the hierarchies of classes of geometric intersection graphs. *Electronic Notes in Discrete Mathematics*, 54:223–228, 2016.
3. J. Cardinal, S. Felsner, T. Miltzow, C. Tompkins, and B. Vogtenhuber. Intersection graphs of rays and grounded segments. In *Graph-Theoretic Concepts in Computer Science (WG 2017)*, volume 10520 of *Lecture Notes in Computer Science*, pages 153–166. Springer, 2017.
4. J. Chalopin and D. Gonçalves. Every planar graph is the intersection graph of segments in the plane: extended abstract. In *ACM Symposium on Theory of Computing, STOC 2009*, pages 631–638. ACM, 2009.
5. P. Damaschke. The Hamiltonian circuit problem for circle graphs is NP-complete. *Inf. Process. Lett.*, 32(1):1–2, 1989.
6. M. Derka. *Restricted String Representations.* PhD thesis, David R. Cheriton School of Computer Science, 2017.
7. G. Ehrlich, S. Even, and R. E. Tarjan. Intersection graphs of curves in the plane. *J. Comb. Theory, Ser. B*, 21(1):8–20, 1976.
8. J. Fox and J. Pach. Separator theorems and Turán-type results for planar intersection graphs. *Adv. Math.*, 219:1070–1080, 2008.
9. J. Fox and J. Pach. A separator theorem for string graphs and its applications. *Combinatorics, Probability & Computing*, 19(3):371–390, 2010.
10. J. Fox and J. Pach. Computing the independence number of intersection graphs. In *ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 1161–1165, 2011.
11. M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM J. Matrix Analysis Applications*, 1(2):216–227, 1980.
12. M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.
13. S. Har-Peled and K. Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. In *23rd Annual European Symposium on Algorithms ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 717–728. Springer, 2015.

14 J. M. Keil, J. S. B. Mitchell, D. Pradhan, and M. Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. *Comput. Geom.*, 60:19–25, 2017. Appeared also in the Proceedings of CCCG 2015.

15 J. Kratochvíl. String graphs II. Recognizing string graphs is NP-hard. *J. Comb. Theory, Ser. B*, 52(1):67–78, 1991.

16 J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *J. Comb. Theory, Ser. B*, 53(1):1–4, 1991.

17 J. R. Lee. Separators in region intersection graphs. In *Innovations in Theoretical Computer Science, ITCS'17*, 2017.

18 J. Matoušek. Near-optimal separators in string graphs. *CoRR*, abs/1302.6482, 2013.

19 M. Middendorf and F. Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Mathematics*, 108(1-3):365–372, 1992.

20 M. Middendorf and F. Pfeiffer. Weakly transitive orientations, Hasse diagrams and string graphs. *Discrete Mathematics*, 111(1-3):393–400, 1993.

21 J. Pach and G. Tóth. Recognizing string graphs is decidable. *Discrete & Computational Geometry*, 28(4):593–606, 2002.

22 M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs is in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

23 M. Schaefer and D. Stefankovic. Decidability of string graphs. In *ACM Symposium on Theory of Computing*, pages 241–246. ACM, 2001.

24 F. W. Sinden. Topology of thin film rc-circuits. *Bell System Technical Journal*, 45:1639–1662, 1966.

25 A. Suk. Coloring intersection graphs of x-monotone curves in the plane. *Combinatorica*, 34(4):487–505, 2014.