FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Martin Töpfer

Intersection representations of graphs

Computer Science Institute of Charles University

Supervisor of the master thesis: doc. RNDr. Vít Jelínek, Ph.D.
Study programme: Computer Science
Study branch: Discrete Models and Algorithms

Prague 2019

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague, May 9, 2019 signature of the author

I would like to thank my supervisor doc. RNDr. Vít Jelínek, Ph.D. for his advice and his interest in the topic. I am especially grateful for his prompt responses and constructive suggestions. I would like to thank my schoolmates from KAM and my co-organizers of Mathematical Correspondence Seminar (PraSe) for keeping me motivated during my studies. Last but not least, I would like to thank my family for their support.

Title: Intersection representations of graphs

Author: Martin Töpfer

Institute: Computer Science Institute of Charles University

Supervisor: doc. RNDr. Vít Jelínek, Ph.D., Computer Science Institute of Charles University

Abstract: This thesis is devoted to the outer and grounded string representations of graphs and their subclasses. A string representation of a graph is a set of strings (bounded continuous curves in a plane), where each string corresponds to one vertex of the graph. Two strings intersect each other if and only if the two corresponding vertices are adjacent in the original graph. An *outer string graph* is a graph with a string representation where strings are realized inside a disk and one endpoint of each string lies on the boundary of the disk. Similarly, in case of *grounded string graphs* the strings lie in a common half-plane with one endpoint of each string on the boundary of the half-plane. We give a summary of subclasses of grounded string graphs and proves several results about their mutual inclusions and separations. To prove those, we use an order-forcing lemma which can be used to force a particular order of the endpoints of the string on the boundary circle or boundary line. The second part of the thesis contains proof that recognition of outer string graphs is NP-hard.

Keywords: outerstring graph, NP-hardness, grounded representation, intersection graph, intersection representation

Contents

Introduction

A string representation of a graph G is a map of its vertices to strings (bounded continuous curves) in a plane such that two strings intersect each other if and only if corresponding vertices are connected by an edge. Such graphs are called *string graphs*. The class of string graphs is one of the oldest and most general graph classes. The concept was firstly used by Benzer [3] in the context of genetic structures and then the class was formalized by Sinden [25] to describe electrical networks and printed circuits. The most well-known results about string graphs are that their recognition is NP-hard Kratochvíl [15] and much later it was proven by Schaefer et al. [23] that it is in fact NP-complete.

One of the main motivations for defining and studying graph classes is finding more efficient algorithms for graphs from that class. Unfortunately, string graphs seem to be too general as there are no positive results about problems that are NP-hard for general graph but easier when restricted to string graphs.

In 1991, Kratochvíl [14] defined *outerstring graphs* as string graphs where strings are contained in a disk and one endpoint of each string is on the boundary of the disk.

A further natural restriction of outerstring graphs is done via geometric restrictions on the shape of strings. We might want strings in the shape of halflines, straight-line segments, axis-aligned L-shapes, etc. This leads to several classes of graphs that have been studied in recent years.

Probably the most relevant paper, where these problems are addressed, is written by Cardinal et al. [5]. Apart from defining several new classes of intersection graphs, they prove their mutual containment and separation. The main tool for that is a lemma that can force the order of strings in an outer or grounded string representation. We generalize this lemma to string representations, where a pair of strings can intersect each other multiple times, and use it for our main results.

Minimum Weight Independent Set (MIS) is an NP-hard problem for general graphs. Recently, outerstring graphs (and their subclasses) received a big attention thanks to the algorithm for MIS problem running in time $\mathcal{O}(k^4)$ for a given a piece-wise linear outerstring representation[1] Keil et al. [13]. Parameter k denotes here the number of total bends in the given piece-wise linear outerstring representation.

In Section 2 we give an overview of several classes of intersection graphs and prove that there are no non-trivial inclusions among them and that all the classes are distinct.

In Section 3 we prove that recognition of outerstring graphs is NP-hard. Our result also implies that even for a graph that has a piece-wise linear outerstring representation with linear (in the number of vertices) number of total bends, the problem of finding its outerstring representation is NP-hard. Therefore to keep the MIS algorithm polynomial, the outerstring representation of the graph must be provided as an input and can not be computed.

[1] If a graph has an outerstring representation, we can modify it such that each string is piece-wise linear. Note that there exist outerstring graphs that require an exponential number of bends in their piece-wise linear outerstring representation.

Part of the thesis from Section 2 (weaker version of Cycle Lemma 1 and Theorems 3, 4 and 5) were previously published as a preprint [12] written by me and my supervisor Vít Jelínek. Because these are our original results obtained while working on the master thesis, we don't mention the citation of [12] separately for every result already published in the preprint.

1. Preliminaries

A graph $G = (V_G, E_G)$ is a simple undirected graph with the set of vertices V_G and the set of edges E_G. For brevity, we will denote edge between vertices u and v as uv instead of $\{u, v\}$.

A graph $H = (V_H, E_H)$ is an *induced subgraph* of graph $G = (V_G, E_G)$ by vertices $V' \subseteq V_G$ if $V_H = V'$ and $E_H = \{uv | uv \in E_G, u \in V', v \in V'\}$. We use the notation that $H = G[V']$.

An *intersection representation* of the graph G is a map of vertices $v \in V_G$ to sets s_v such that two sets s_v and s_u intersect if and only if the corresponding vertices v and u are adjacent in G. Graph G it then called an *intersection graph* of the set $\{s_v | v \in V_G\}$.

A *string representation* of the graph G is an intersection representation where s_v are bounded continuous curves (also called *strings*) in the plane. Graphs, for which exist their string representations, are called *string graphs* and we denote the class of string graphs as String.

We can further restrict the class of String graphs by restricting where the strings can be placed.

An *outer* representation of the graph G is a string representation of G where all strings are realized inside a disk and one endpoint of each string lies on the boundary B of the disk. Graphs with such representation are called *outerstring graphs* or *outer string graphs* and we denote their class as Outer-string. The endpoint of a string, that is on the boundary B, is called *anchor*.

Similarly, a *grounded* representation of the graph G is a string representation of G where each string has one endpoint on a common line called *grounding line* and the rest of the string belongs to one halfplane defined by the grounding line. For easier visualization, we will consider the grounding line to be horizontal and the halfplane, that contains strings, to be below that line. The endpoints of strings on the grounding line are called *anchors*.

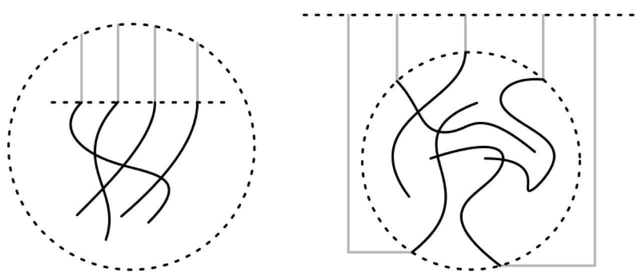

Figure 1.1: An example of grounded (left) and outer (right) string representation in black color. The gray lines show a reduction from grounded to outer representation and vice versa. Adopted from Cardinal et al. [5].

It is easy to see that in the case of general strings a graph admits an outerstring representation if and only if it admits a grounded representation (Cardinal et al. [5], see Figure 1.1). The reason for separate definitions is that with additional constraints on the shape of strings these classes may differ.

We say that a graph class $\mathcal{C}_1$ is a *proper subclass* of $\mathcal{C}_2$ if $\mathcal{C}_1 \subseteq \mathcal{C}_2$ and $\mathcal{C}_1 \neq \mathcal{C}_2$

In Section 3 we are concerned about the computational complexity of recognition of outerstring graphs. NP is a class of decision problems solvable in polynomial time by a non-deterministic Turing machine. A problem P is *NP-hard* if every problem Q from NP can be reduced in polynomial time to P. To show that a problem P is NP-hard is enough to show a polynomial reduction of any NP-hard problem to P.

The most well-known NP-hard problem is SAT. A logical formula is in *conjunctive normal form* (CNF) if it is a conjunction of *clauses*, where a clause is a disjunction of *literals*. The SAT problem is a decision problem whether there exists a satisfying assignment of variables of given CNF formula or not.

Definition 1. *An* exact-3-CNF *formula is a logical formula in conjunctive normal form where each clause consists of exactly three distinct literals.*

Definition 2. *An* Exact-3-SAT problem *is a problem of deciding whether a given exact-3-CNF formula have a satisfying assignment or not.*

It is a well-known fact that Exact-3-SAT, as well as many other variants of SAT problem, is NP-hard. This can be shown by a trivial reduction from the NP-hardness of the SAT problem.

2. Classes of intersection graphs

Here we mention some well-known graph classes and relations between them. The summary of all inclusions among the classes we are later more interested in is given in Figure 2.1. Additional arguments that there are no missing inclusions are provided in Section 2.2. Our original results are:

- GROUNDED-$\{\mathsf{L}, \mathsf{⅃}\}$ $\nsubseteq$ GROUNDED-L (Theorem 3)
- GROUNDED-SEG $\nsubseteq$ GROUNDED-$\{\mathsf{L}, \mathsf{⅃}\}$ (Theorem 4)
- MPT $\nsubseteq$ OUTER-1-STRING (Theorem 5)

In the following list of graph classes is in parenthesis mentioned notation for the given graph class that will be used in the rest of the thesis.

k-string Graphs (k-STRING) A string graph G is a *k-string graph* if each pair of strings has at most k common intersections. Restricting number of intersections restricts graphs in the class because there exist string graphs requiring exponential number of intersections [16].

Outer-k-string graphs (OUTER-k-STRING) A graph G is *outer-k-string graph* if it has an outerstring representation where each pair of strings has at most k common intersections. Note that OUTER-k-STRING $\neq$ OUTER-STRING $\cap$ k-STRING.

Piece-wise linear outerstring graphs Each outerstring representation of graph G can be transformed to a representation where each string is replaced by a piece-wise linear curve. Because the class of such graph is exactly OUTER-STRING, this notion is useful just for algorithmic purposes to parametrize time complexity of an algorithm by number of straight line segments k in this representation.

SEG graphs (SEG) Intersection graphs of straight segments are called *segment graphs* or just *SEG graphs*. After several results about subclasses of planar graphs being in SEG, in 2019 Chalopin and Gonçalves [7] finally proved the Scheinerman's conjecture from 1984, that every planar graph admits a SEG representation. Recognition of segment graphs is known to be NP-hard [17]. More recently Matoušek [18] showed that recognition of segment graphs is complete in the existential theory of the reals[1]. Another quite recent result about SEG graphs is that they are are not χ-bounded [21], which means that the chromatic number of the graphs is bounded from above by a function of their clique number.

B_k-VPG graphs (B_k-VPG) Vertex intersection graphs of paths on a grid (VPG) are intersection graphs of strings that are formed of several consecutive horizontal or vertical line segments. B_k-VPG graphs are subclass of VPG graphs, where each string may contain at most k bends. They were introduced by Asinowski et al. [2][2] but the motivation for such class goes back to the first paper

[1]The existential theory of the reals is a problem known to be NP-hard that lies in PSPACE.

about intersection graphs regarding RC circuits. A folklore result is that VPG graphs are exactly String graphs. In Chaplick et al. [9] was given a complete hierarchy of VPG graphs – for every k is B_k-VPG a proper subclass of B_{k+1}-VPG. In the same paper was shown that recognition of B_k-VPG graphs is NP-hard and that for $k \geqslant 1$ there is no inlcusion relation between Seg and B_k-VPG graphs. In recent years there were also several results about how restricted subclass of B_k-VPG still contain planar graphs, for example Chaplick and Ueckerdt [8] proved that planar graphs are B_2-VPG graphs.

L-graphs (L) L-graphs are intersection graphs of axis-aligned L-shapes. By L-shape we mean a union of a horizontal and a vertical segment, in which the left endpoint of the horizontal segment and the bottom endpoint of the vertical segment coincides. We can look at L-graphs as a further restricted subclass of B_1-VPG graphs. There was a conjecture for a long time that planar graphs are even subclass of L-graphs, which was finally proven by Gonçalves et al. [11]. Since it is known that L-graphs are a subclass of segment graphs [20], this result strengthen a previous result that all planar graphs are segment graphs by Chalopin and Gonçalves [7].

Max point-tolerance graphs (Mpt) There are two common names for the class of intersection graphs of L-shapes where all L-shapes have their bends on a common downward-sloping line – they are called either *Max point-tolerance graphs* or *Monotone L-graphs*. This happened because the class was independently introduced by Catanzaro et al. [6] and by Ahmed et al. [1]. We will mostly use the shortcut version of the former name – Mpt. Alternative characterization of this class is via graph that admit a vertex ordering that avoids a certain forbidden pattern [1, 6]. This graph class is also known to contain several well-known subclasses, such as outerplanar graphs and interval graphs [1, 6].

Outer segment graphs (Outer-seg**)** Outer segment graphs are graphs with an outer representation of straight-line segments. This class was introduced by Cardinal et al. [5] where was proven that it is a proper subclass of Outer-1-string. The recognition of Outer-seg is complete in the existential theory of the reals.

Ray graphs (Ray**)** Another class defined in [5] are ray graphs which are intersection graphs of rays (=halflines) in a plane. The two notable results are that Ray is proper subclass of Outer-seg and that their recognition is $\exists\{R\}$-complete [5].

Grounded segment graphs (Grounded-seg**)** Grounded segment graphs are graphs admitting a grounded string representation where each string is a line segment. The class of intersection graphs of downward rays (Down-ray) is exactly Grounded-seg [5]. In the same paper it was shown that grounded segment graphs are proper subclass of ray graphs.

Grounded L-graphs (GROUNDED-L) Grounded L-graphs are the intersection graphs of grounded L-shapes, that is, L-shapes with top endpoint on the horizontal grounding line. We get the same class if we consider outerstring graphs with strings of L-shapes that have their upper endpoint of the horizontal segment placed on the boundary B of the disk. This class has been first considered by McGuinness [19] who has shown that this class is χ-bounded. The χ-boundedness result has been later generalized to all outerstring graphs by Rok and Walczak [22].

Grounded $\{\mathsf{L}, \lrcorner\}$-graphs (GROUNDED-$\{\mathsf{L}, \lrcorner\}$) The class of grounded $\{\mathsf{L}, \lrcorner\}$-graphs is similar to GROUNDED-L, but the representation may use both L-shapes and ⅃-shapes. A ⅃-shape consists of a horizontal and a vertical segment which are placed such that the right endpoint of the horizontal segment coincides with the bottom endpoint of the vertical segment. Middendorf and Pfeiffer [20] showed that GROUNDED-$\{\mathsf{L}, \lrcorner\}$is a subclass of GROUNDED-SEG by vertical stretching of the grounded segment representation such that the segments locally behaves almost as horizontal segments of L-shapes and ⅃-shapes.

Circle graphs (CIRCLE) Circle graphs are the intersection graphs of chords inside a circle. By firstly shifting all endpoints of chords into the upper right quarter of the circle and then changing straight-line chords into L-shapes we can get equivalent intersection representation by L-shapes drawn inside a circle, so that both endpoints of each L-shape touch the circle [2]. Circle graphs include all outerplanar graphs [26].

Interval graphs (INT) Interval graphs are the intersection graphs of intervals on the real line. It is easy to see that interval graphs are subclass of GROUNDED-L. Interval graphs are equivalent to grounded-L graphs in which the length of horizontal segments of L-shapes increases from left to right. Similarly, interval graphs are also subclass of MPT because they are equivalent to MPT representations where all vertical segments reaches above the highest anchor. Note that not every graph from the intersection of MPT and GROUNDED-Lis an interval graph. Any cycle C_n of length $n \geqslant 4$ is a counterexample.

Permutation graphs (PER) Permutation graphs are the intersection graphs of line segments whose endpoints lie on two parallel lines. Equivalently, we may observe that these are exactly the graphs admitting an L representation in which the top endpoints of all the L-shapes are on a common horizontal line and the right endpoints are on a common vertical line.

Outerplanar graphs (OUTERPLANAR) Outerplanar graphs are graphs that have a planar drawing where all vertices belong to the outer face of the drawing.

For simplicity we will assume in this thesis that certain degenerative cases are not allowed. We forbid self-intersection of strings and infinite many points of intersection of two strings (except interval graphs). We also assume that when two strings share a common point, it must be a proper crossing (i.e. strings can not just touch each other). Moreover, we assume that L-shapes are not degenerated

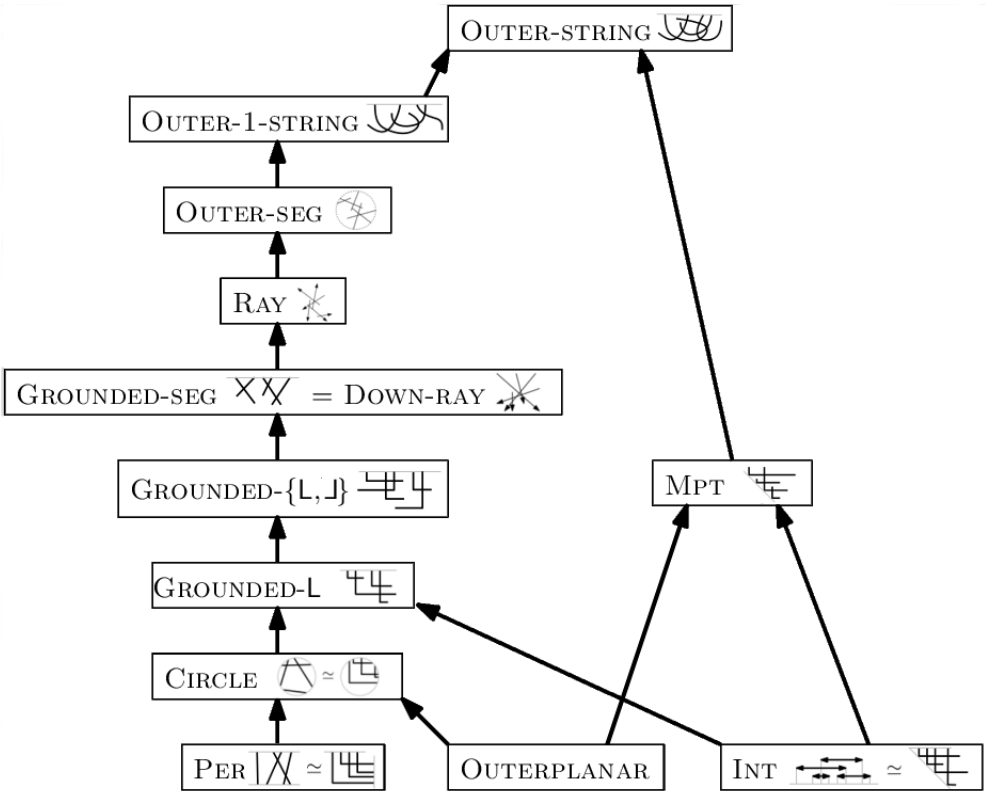

Figure 2.1: Some notable graph classes. We will show that there are no inclusions apart from those represented by arrows and and those following from transitivity. Moreover, all the graph classes are distinct.

which means that the left and the right endpoints of the horizontal segment in the L-shape are distinct and similarly the top and the bottom endpoints of the vertical segment in the L-shape are distinct. These assumption doesn't affect the expressive power of most intersection graph classes but they allow us to make our arguments more clear because we don't need to discuss corner cases. The only affected classes are the k-STRING and OUTER-k-STRING where the forbidden touching of two strings instead of proper crossing might increase number of intersections needed in the string representation.

2.1 Forcing induced order of vertices

For each grounded representation of a graph G with a horizontal grounding line, the left-to-right order of anchors on the grounding line defines a linear order of the vertices V_G that correspond to the anchors. We say that this order is *induced* by the grounded representation. Similarly, for MPT representations we define the induced order of vertices V_G as the order of the bends of L-shapes, that correspond to these vertices, on the common sloping line.

The induced order of an outerstring representations of G is the clockwise order of anchors of strings, that correspond to the vertices, on the boundary B of the disk from top of the disk. Because each outerstring representation can be rotated or inverted without affecting the outerstring representation in any way, we define that such orders are equivalent.

Formally, let G be a graph with vertices $V_G = \{x_1, x_2, \dots, x_n\}$ and $<$ be linear order of its vertices $x_1 < x_2 < \cdots < x_n$. We will call an *ordered graph* $(G, <)$ the pair of a graph G with a linear order of its vertices $<$. We say that two orders are *equivalent* if we can get one order from the other by a finite sequence of *cyclic shifts* ($x_1 < x_2 < \cdots x_n \to x_2 <_c \cdots <_c x_n <_c x_1$) and *reversals* ($x_1 < x_2 < \cdots x_n \to x_n <_r < \cdots <_r x_2 <_r < x_1$).

We say that graph G admits a *constrained outerstring representation* with the order $<$ if there exists an outerstring representation of G that induces $<$.

Now we are ready to formulate the Cycle Lemma which is used to force any induced order (or order equivalent to that order) in an outerstring representation of the graph G. We present a generalization of Cycle Lemma by Cardinal et al. [5] from graphs with an outer-1-string representation to graphs with outerstring or grounded string representation and to several additional graph classes.

A *cycle extension* of the ordered graph $(G, <_v)$ is an unordered graph $H = (V_H, E_H)$ with these properties (see Figure 2.2):

- V_H is the disjoint union of the sets $V_G = \{x_1, \dots, x_n\}$ and $V_C = \{y_1, \dots, y_{5p}\}$.
- Vertices of V_G induce a copy of G and vertices of V_C induce a cycle of length $5p$ with edges $y_1y_2, y_2y_3, \dots, y_{5p-1}y_{5p}, y_{5p}y_1$.
- Each vertex $x_i \in V_G$ is either adjacent to y_{5p} and has no other neighbors in V_C, or is adjacent to y_{5p} and y_{5p-1} and has no other neighbors in V_C

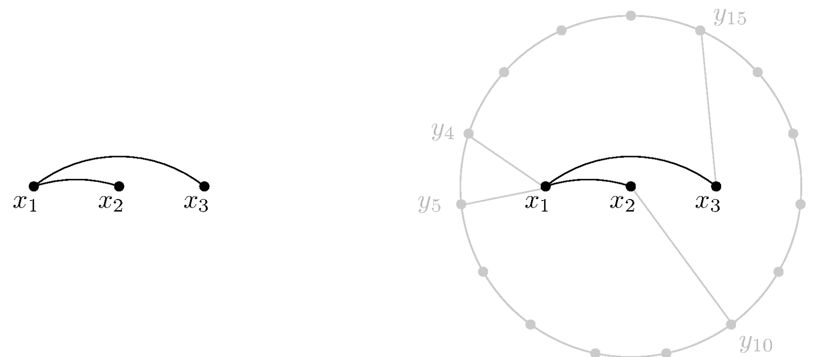

Figure 2.2: An ordered graph $(G, <)$ (left) and one of its possible cycle extensions H (right).

Lemma 1 (Cycle Lemma). *Let $(G, <)$ be an ordered graph with cycle extension H defined as above. Then in every grounded string representation and in every outerstring representation of H is the order of vertices V_G induced by the representation equivalent to the order $<$.*

*On the other hand, for any graph class $\mathcal{C} \in \{$*Grounded-L, Grounded-{L, ⅃}, Mpt, Grounded-seg, Outer-1-string, Outer-string*$\}$, for every $\mathcal{C}$ representation of a graph G inducing an order $<$ on V_G there is a cycle extension H of $(G, <)$ such that a $\mathcal{C}$ representation of H can be constructed by adding into the given representation of G the curves representing the vertices of $V_H \backslash V_G$.*

Before proving Lemma 1 we will study outerstring representations of cycles. Firstly, we need a little bit of notation. Let s be a string and $X = \{x_2, x_3\}$ be set of some important points on the string s. By $s(x_2, x_3)$ we will denote part of the string s between points x_2 and x_3. Let x_1 be an endpoint of s. Then by the *nearest* point to x_1 in s we mean the point which we will visit first if we start in the point x_1 and move along the string s. Formally the nearest point to x_1 in s from points X is such $x_i \in X$ that $s(x_1, x_i) \cap X = \{x_i\}$.

Lemma 2. *Let C be a cycle on vertices $y_1, y_2, \ldots, y_{5n}$ with an outerstring representation where a vertex y_i is represented by a string c_i. Then there exists a closed Jordan curve which is a union of substrings center(i) of strings c_i. The order of the substrings on J is equivalent to center(1) < center$(2) < \cdots <$ center$(5n)$.*

Proof. Denote a_i the anchor of string c_i in the outerstring representation and a'_i the second endpoint of string c_i.

We will describe the construction of the closed Jordan curve J.

Denote $p'_{1,2}$ an arbitrary intersection of c_1 and c_2. The point $p'_{1,2}$ divides c_2 into two distinct subcurves $c_2(a_2, p'_{1,2})$ and $c_2(p'_{1,2}, a'_2)$. We pick the subcurve that contain some intersection with c_3. If both of them intersect c_3, we can pick arbitrarily. We denote the nearest intersection with c_3 as $p_{2,3}$. We continue this process until we get back to the intersection with c_1 and define point $p_{5n,1}$. Formally, by "continuing this process" we mean that for $i \in \{3, 4, \ldots, 5n\}$ we perform the following step: In step i we start at point $p_{i-1,i}$, pick one of the subcurves $c_i(a_i, p_{i-1,i})$, $c_i(p_{i-1,i}, a'_i)$ that intersect c_{i+1} and denote $p_{i,i+1}$ the nearest intersection with c_{i+1} from $p_{i-1,i}$ along the picked subcurve.

Now it remains to define $p_{1,2}$. If there is no intersection of $c_1(p_{5n,1}, p'_{1,2})$ and $c_2(p'_{1,2}, p_{2,3})$, we rename $p'_(1, 2)$ to $p_{1,2}$. Otherwise we denote $p_{1,2}$ the nearest intersection with $c_2(p'_{1,2}, p_{2,3})$ from $p_{5n,1}$ in $c_1(p_{5n,1}, p'_{1,2})$. See Figure 2.3.

At the end of this process we call *central part* of c_i, denoted as center(i), the subcurve of c_i between $p_{i-1,i}$ and $p_{i,i+1}$. By taking the union over all central parts center(i) we get a closed Jordan curve J.

The fact, that J is non-self-intersecting follows from its construction and can be proved by mathematical induction: We start with $J = \{\}$ and just strings s_1 in the representation. Then we will be adding central parts center(i) with corresponding strings s_i for $i \in \{2, 3, \ldots, 5n\}$. When we add central part center(i), it can potentially intersect only the previous part center$(i-1)$ because s_i doesn't intersect any other already added strings (string s_{i+1} will be added later). But intersection of center$(i-1)$ and center(i) would contradict the choice of $p_{i-1,i}$ as the nearest intersection of the two strings. Now remain just the last central part – center(1). Because of the possible change of $p'_{1,2}$ to the nearest intersection of center(1) with center(2) from $p_{5n,1}$, center(1) can not intersect center(2). The order of center(i) strings in J immediately follows from the construction. □

Proof of Theorem 1. Let us begin with the first part. Let $(G, <)$ be an ordered graph with a cycle extension H as described above. For convenience we will treat indices of vertices from V_C modulo $5n$ (so $y_{5p+1} = y_1$ etc.).

If $(G, <)$ has a grounded string representation we can transform it to an outerstring representation by placing the boundary B of the disk around the whole grounded representation and prolonging the strings to the boundary without crossing each other (see Figure 1.1).

Let s_i be a string corresponding to the vertex $x_i \in V_G$, c_i a string corresponding to $y_i \in V_C$, a_i the anchor of c_i and a'_i the other endpoint of c_i.

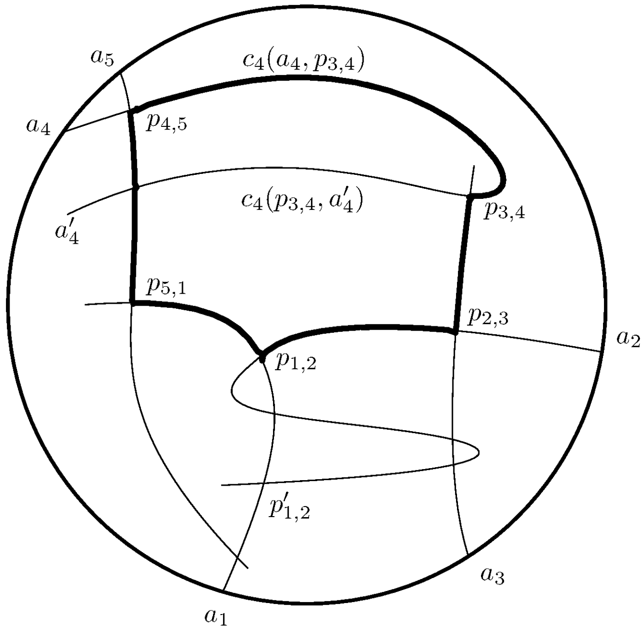

Figure 2.3: An outerstring representation of 5-cycle. The resulting Jordan curve J is thickened. Note that at the beginning it would be hard to determine the correct position of $p_{1,2}$ as it depends on the position of $p_{5,1}$. In case of the depicted graph it would be sufficient to study c_5 but for more complex outerstring representations with more intersections we might need to analyze more strings. In extreme case the, choice might even depend on the position of $p_{1,2}$.

According to Lemma 2, there exists a closed Jordan curve J, which is a union of so-called *central parts* $\text{center}(i) \subseteq c_i$ that are in J in the order equivalent to $\text{center}(1) < \text{center}(2) < \cdots < \text{center}(5n)$. Further we will use the same notation as is used in the lemma.

Denote the first point of c_i from anchor a_i contained in J as p_i. The *initial part* of a string c_i, denoted as $\text{start}(c_i)$, is the part of c_i from its anchor a_i to p_i. Let R_J be the interior region of J.

Let a, b, c be three distinct points on the Jordan curve J. Then denote $\text{curve}(a, b, c)$ the part of J bounded by points a and c which contain b. Similarly, for distinct points $a, b, c \in B$ let $\text{bound}(a, b, c)$ be the part of the boundary B bounded by points a, c containing point b. We define R_i as the interior region of Jordan curve that we get as a union of $\text{start}(5i - 3)$, $\text{curve}(p_{5i-3}, p_{5i}, p_{5i+2})$, $\text{start}(5i + 2)$ and $\text{bound}(a_{5i+2}, a_{5i}, a_{5i-3})$ (see Figure 2.4).

Note that string s_i is the only string corresponding to some vertex from V_G that can intersect the boundary of R_i. String c_{5i} is contained in $R_i \cup R_J$ because part of the string is inside the union ($\text{center}(5i)$ is part of the boundary of R_i) and c_{5i} cannot intersect the boundary of $R_i \cup R_J$. Because s_i must intersect $c_5 i$, at least part of s_i is also in $R_i \cup R_J$. Because s_i cannot intersect the boundary of $R_i \cup R_J$, the whole string is contained in $R_i \cup R_J$. In particular, the anchor of s_i is in $R_i \cap B$. Therefore, the anchors of $s_1, s_2, \ldots, s_n$ appear on B in the order which, up to equivalence, corresponds to the order $<$ on V_G.

The second part of the proof is much easier. Let us assume that we are given a $\mathcal{C}$ representation of G. It is easy to see that for all mentioned graph classes we can realize the new vertices of cycle extension in small enough neighborhood of

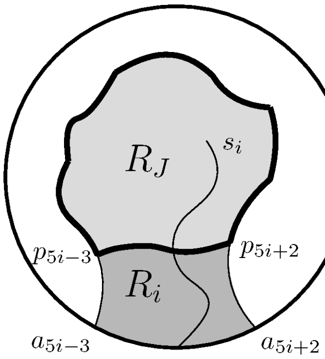

Figure 2.4: Regions R_i and R_J for a string s_i. The thickened line is the Jordan curve J. String s_i cannot leave the region $R_i \cup R_J$.

grounding line or boundary B of the disk. In case of Mpt representation we use a cycle extension where each vertex from V_G is adjacent to two vertices y_{5i-1} and y_{5i} from V_G. In case of all other classes we use a cycle extension where each vertex from V_G is adjacent only to one vertex y_{5i} from V_G. For examples of extensions of the $\mathcal{C}$ representations see Figure 2.5. $\square$

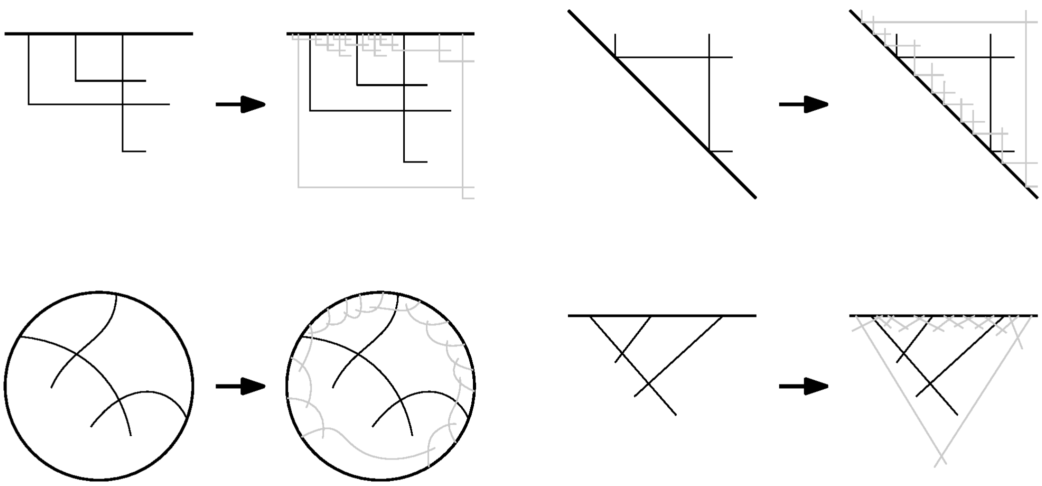

Figure 2.5: Extending the representation of G into a representation of its cycle extension for Grounded-L, Mpt, Grounded-seg and outer-1-string representations. Note that for Grounded-$\{\mathsf{L}, \lrcorner\}$ representations we can use the same approach as for Grounded-L. Similarly, the extension of Outer-1-string can be used for Outer-string representations.

2.2 Separation between graph classes

Our goal is this section is to show that there are no inclusions among classes from Figure 2.1 apart from the depicted ones and that all the classes are distinct.

The classes Int, Circle, Outerplanar and Per are well studied and it is known that the only inclusions among them are Outerplanar $\subsetneq$ Circle and Per $\subsetneq$ Circle.

The upper-right part of Figure 2.1 was proven by Cardinal et al. [5]. In fact, several of those classes were firstly defined in that paper and the proper inclusions among them are one of a few known results about those classes.

$$\textsc{Grounded-seg} = \textsc{Down-ray} \subsetneq \textsc{Ray} \subsetneq \textsc{Outer-seg} \subsetneq \textsc{Outer-1-string}$$

Catanzaro et al. [6, Observation 6.9] showed that the graph $K_{2,2,2}$ (the octahedron) is a permutation graph while it is not in Mpt, and therefore no superclass of Per is contained in Mpt. On the other hand, we will show in Theorem 5 that there exists a graph from Mpt not contained in Outer-1-string. Thus Mpt is incomparable with all the classes that are supersets of Per and subsets of Outer-1-string. Because both Outerplanar and Int are subsets of Mpt but they are incomparable between each other, none of these classes can be a superset of Mpt. By using the same argument for Mpt and Outer-1-string, which are incomparable between each other, we get that both these classes are proper subclasses of Outer-string. And similarly also both Circle and Int are proper subclasses of Grounded-L. To complete the hierarchy, we only need the separations from Theorem 3 and Theorem 4.

For the following proofs we will use Cycle Lemma to show that some graph doesn't belong to certain graph class. The ability to prescribe order of the vertices enable us to come up with relatively small examples which are easy to analyze.

Theorem 3. *The class* Grounded-L *is a proper subclass of* Grounded-{L, ⅃}.

Proof. From definition of Grounded-L and Grounded-{L, ⅃} graphs follows that Grounded-L⊆Grounded-{L, ⅃}. To show that these classes are distinct, consider graph $G = (V_G, E_G)$ with vertices $V_G = \{x_1, x_2, x_3, x_4\}$ and edges $E_G = \{x_1x_2, x_2x_3, x_1x_4\}$. Figure 2.6 shows a grounded-{L,⅃} representation of G which induces the order of vertices $x_1 < x_2 < x_3 < x_4$.

Let us show that the graph G doesn't have a grounded-L representation with the induced order of the vertices $x_1 < x_2 < x_3 < x_4$. This can be simply argued from characterization of Grounded-L graphs by forbidden patterns by Jelínek and Töpfer [12] because one of the forbidden patterns is exactly the graph G.

For the sake of completeness we include an alternative short prove here: Let ℓ_i denote an L-shape corresponding to the vertex x_i and h_i denote the length of the vertical segment in the L-shape ℓ_i. Because $x_1x_4 \in E_G$, the vertical segment of ℓ_1 reaches all the way to the right to ℓ_4. Because $x_1x_2 \in E_G$, the vertical segment of ℓ_2 is below the vertical segment of ℓ_1 ($h_1 < h_2$). Because $x_1x_3 \notin E_G$, $h_3 < h_1$. That is a contradiction as there is no way how to realize the edge x_2x_3. See Figure 2.6 for illustration.

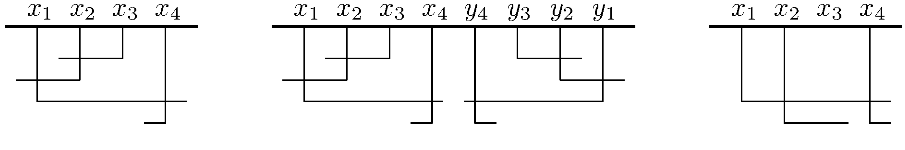

Figure 2.6: Grounded-{L,⅃} representation of graphs G and G' as defined in the proof of Theorem 3 (G left, G' middle) and a partial grounded-L representation of G that can not be extended to the full representation (right).

Let $(G', <_{G'})$ be an ordered graph obtained by putting graph G and a mirror image of G next to each other. Formally, $(G', <_{G'})$ is an ordered graph with

$V_{G'} = \{x_1, x_2, x_3, x_4, y_4, y_3, y_2, y_1\}$, $E_{G'} = \{x_1x_2, x_2x_3, x_1x_4, y_1y_2, y_2y_3, y_1y_4\}$ and vertex order $<_{G'}$ defined as $x_1 <_{G'} x_2 <_{G'} x_3 <_{G'} x_4 <_{G'} y_4 <_{G'} y_3 <_{G'} y_2 <_{G'} y_1$. Now let $(G'', <_{G''})$ be an ordered graph obtained by putting two distinct copies of $(G', <_{G'})$ next to each other.

We apply Cycle Lemma 1 to the ordered graph $(G'', <_{G''})$. The second part of the lemma says that there exists a grounded-{L,⅃} representation of graph H, which is a cycle extension of the ordered graph $(G'', <_{G''})$. Let G^* be an induced subgraph of H by vertices $V_{G''}$. The first part of the lemma states that any grounded string representation of G^* induces an order of the vertices equivalent to $<_{G''}$. From construction of the graph G'' follows that there exist four consecutive vertices z_1, z_2, z_3, z_4 in V_{G^*} that induces exactly the ordered graph $(G, <)$. Such graph doesn't have a grounded-L representation, thus also H doesn't admit a grounded-L representation. It follows that Grounded-L≠Grounded-{L, ⅃}. □

In similar way we prove that also other classes of graphs are distinct. We only need to pick a different initial ordered graph $(G, <)$ that doesn't belong to the smaller class.

Theorem 4. *The class* Grounded-{L, ⅃} *is a proper subclass of* Grounded-seg.

Proof. It was proven by Middendorf and Pfeiffer [20] that Grounded-{L, ⅃} is a subclass of Grounded-seg. To show that these classes are distinct, consider graph $G = (V_G, E_G)$ with vertices $V_G = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and edges $E_G = \{x_1x_2, x_1x_3, x_1x_5, x_1x_6, x_2x_5, x_2x_6, x_4x_6, x_5x_6\}$. Figure 2.7 shows a grounded-SEG representation of G which induces the order of vertices $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$.

Let us show that the graph G doesn't have a grounded-{L,⅃} representation with the induced order of the vertices $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$. For contradiction, let us assume that there is such representation. Denote ℓ_i an L-shape or ⅃-shape corresponding to x_i in the representation. Let h_i denote the length of the vertical segment of ℓ_i. Without loss of generality let us assume that $h_1 < h_6$. Thus ℓ_1 is an L-shape. ℓ_4 intersects ℓ_6 but not ℓ_1 and therefore it is an L-shape with $h_4 < h_1$ (see Figure 2.7). ℓ_5 should intersect both ℓ_1 and ℓ_6 but not ℓ_4 which is impossible. Contradiction.

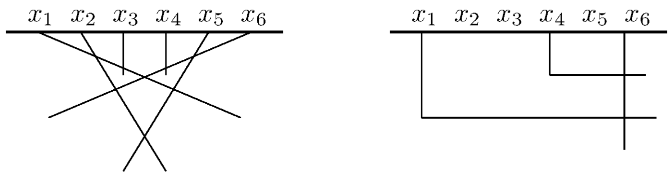

Figure 2.7: Grounded segment representation of the graph G as defined in the proof of Theorem 4 (left) and its grounded-{L,⅃} representation of vertices x_1, x_4 and x_6 (right).

The ordered graph $(G, <)$ is symmetrical, which means that $(G, <)$ is isomorphic to $(G, <_r)$ where $<_r$ is reversal order of $<$ as defined in the Section 2.1. Therefore we don't need to add a mirror image of $(G, <)$. Let $(G'', <_{G''})$ be an ordered graph obtained by putting two distinct copies of $(G, <)$ next to each other.

Let H be a cycle extension of the ordered graph $(G'', <_{G''})$. According to Cycle Lemma 1 H admits a grounded segment representation. Because the order of the vertices from $V_{G''}$ in H is equivalent to $<_{G''}$, there exist six vertices that induce the ordered graph $(G, <)$. Thus $H \notin$ GROUNDED-{L, ⅃}and GROUNDED-L$\neq$ GROUNDED-SEG. □

Theorem 5. *The class* MPT *is not a subclass of* OUTER-1-STRING.

Proof. Let us consider a graph G, whose MPT representation is depicted in the left part of Figure 2.8. Its set of vertices is $V_G = \{x_1, x_2, \ldots, x_7\}$ and its set of edges is $E_G = \{x_1x_5, x_1x_7, x_2x_4, x_3x_4, x_3x_5, x_3x_6, x_4x_5, x_4x_6\}$. Let $<$ be the induced order of its vertices $x_1 < x_2 < \cdots < x_7$.

We claim that the ordered graph $(G, <)$ doesn't have an outer-1-string representation. Denote s_i a string corresponding to x_i, a_i the anchor of s_i and p_{ij} the intersection point of s_i and s_j if x_i and x_j are adjacent. Because each pair of strings has at most one intersection, p_{ij} is well defined. The graph is symmetrical and thus without loss of generality we can assume that s_4 intersects s_2 before it intersects s_6. Let J be a closed Jordan curve consisting of the subcurve of s_1 between a_1 and p_{17}, the subcurve of s_7 between p_{17} and p_{37}, the subcurve of s_3 between p_{37} and a_3 and the segment of the grounding line between a_3 and a_1. Because a_4 and s_6 are outside the closed curve J and the string s_2 is inside the curve (resp. a_2 is on the curve J), s_4 must intersect J twice. However, s_4 can intersect J only in the subcurve of s_3 which forces two intersections of s_4 and s_3. Contradiction.

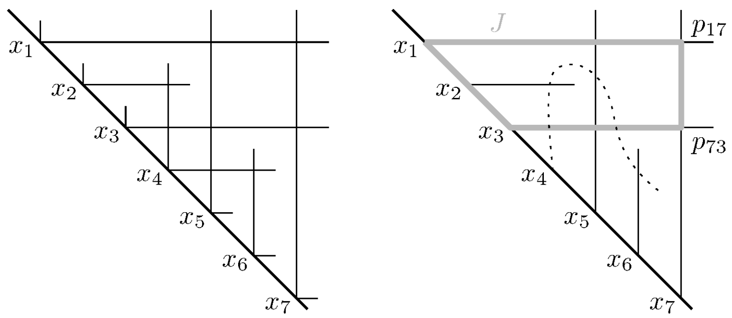

Figure 2.8: MPT representation of the graph G used in the proof of Theorem 5 (left) and highlighted Jordan curve J that s_4 must intersect twice (right).

Let H be a cycle extension of G. From Cycle Lemma 1 follows that H admits an MPT representation but H doesn't have an outer-1-string representation because $H[V_G]$ is an ordered graph isomorphic to $(G, <)$. □

3. Recognition of outerstring graphs is NP-hard

One of the basic and most natural questions about any graph class is how to recognize whether a given graph G belongs to the class and computational complexity of such question. It was shown by Kratochvíl [15] that recognizing string graphs is NP-hard. Later results by Kratochvíl and Matoušek [17] show that the recognition of SEG is complete in the existential theory of the reals. Cardinal et al. [5] later showed that also the recognition of GROUNDED-SEG, OUTER-SEG and RAY is complete in the existential theory of the reals. We continue in a similar direction and prove that the recognition of OUTER-STRING graphs is NP-hard.

Positive complexity results about recognition of intersection graphs include polynomial algorithms for recognizing interval graphs Booth and Lueker [4], circle graphs Gabor et al. [10] and intersection graphs of rays in two directions Shrestha et al. [24].

In this section we show that recognition of OUTER-STRING graphs is NP-hard by reduction from Exact-3-SAT. The main idea of the construction is similar to the construction by Kratochvíl [15] where it was shown that recognition of string graphs is NP-hard.

Theorem 6. *The problem of deciding whether a given graph G is an outer-2-string graph or it is not an outerstring graph is NP-hard.*

3.1 Cycle Lemma for subset-orderings

For creating variable-gadgets we use a generalized Cycle Lemma for graphs which does not prescribe total ordering of their vertices but leaves some freedom by enforcing only some particular partial order.

Definition 3. *A partial order $<_v$ of the vertices of a graph G is a* subset-order *if there exists a partition $X_1, X_2, \dots, X_p$ of vertices V_G with linear ordering $X_1 <_s X_2 <_s \dots <_s X_p$ such that $x_i <_v x_j$ if and only if $x_i \in X_i$, $x_j \in X_j$ and $X_i <_s X_j$*

We say that a linear order $<$ of vertices V_G is *consistent* with a subset-order $<_v$ if $<_v$ can be extended to a linear order equivalent to $<$.

A *subset-cycle extension* of the subset-ordered graph $(G, <_v)$ is an unordered graph $H = (V_H, E_H)$ with these properties:

- V_H is the disjoint union of the sets $V_G = \{x_1, \dots, x_n\}$ and $V_C = \{y_1, \dots, y_{5p}\}$.
- Vertices of V_G induce a copy of G and vertices of V_C induce a cycle of length $5p$ with edges $y_1y_2, y_2y_3, \dots, y_{5p-1}y_{5p}, y_{5p}y_1$
- Each vertex $x_i \in X_j$ is adjacent to y_{5j}.

Lemma 7 (Subset Cycle Lemma)**.** *Let $(G, <_v)$ be a subset-ordered graph and H its subset-cycle extension as defined above. Then $<_v$ is consistent with the order of the vertices V_G induced by any outerstring representation of H.*

On the other hand, let $<$ be an ordering of vertices $V(G)$ induced by an outerstring representation of graph G. Then for any subset-order $<_v$ consistent with $<$ the subset-cycle extension H of $(G, <_v)$ has an outerstring representation which can be constructed from the outerstring representation of G just by adding curves representing the vertices of $V_H \backslash V_G$.

Proof. Let us assume for contradiction that there exists an outerstring representation of H which is not consistent with subset-order $<_v$. That means that we can pick p vertices from V_H such that each of these vertices is from different part of the partition $X_1, X_2, \dots, X_p$ and their order is not equivalent with the order of the partition. More precisely, we denote the picked vertices $z_1, z_2, \dots, z_p$ such that $z_i \in X_i$ for all $i \in \{1, 2, \dots, p\}$. Let $<_z$ be the order of these vertices in the outerstring representation. Because these vertices are not ordered according to the subset order $<_v$, $<_z$ is not equivalent with $<_v$.

A graph induced by vertices $\{z_1, \dots, z_p\} \cup \{y_1, y_2, \dots, y_{5p}\}$ is exactly the cycle extension of an ordered graph $(\{z_1, \dots, z_p\}, <_v)$ according to Lemma 1. We get a contradiction with the fact that $<_v$ and $<_z$ are not equivalent.

On the other hand, for any order of the vertices consistent with $<_v$ we can extend the outerstring representation in the neighborhood of the boundary circle B similarly as in the case of Lemma 1. $\square$

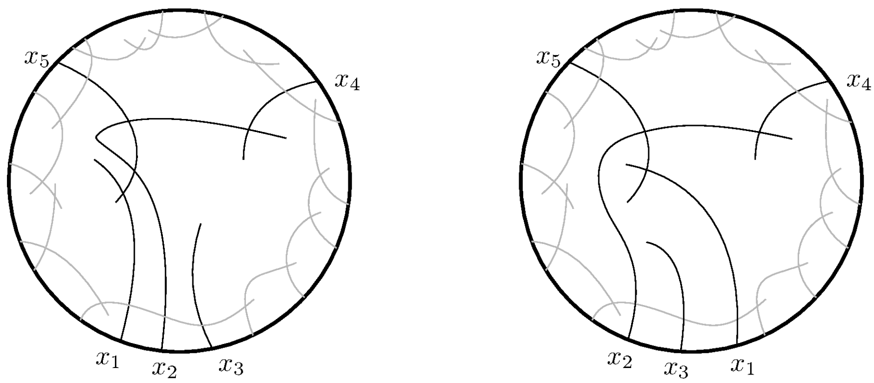

Figure 3.1: Two possible realizations of a subset-ordered graph G and its subset-cycle extension. The subset-order is $\{x_1, x_2, x_3\} < x_4 < x_5$.

3.2 Construction

Let $\phi = c_1 \wedge c_2 \wedge \dots \wedge c_m$ be an exact-3-CNF formula containing variables $x_1, x_2, \dots x_n$ and clauses $c_1, c_2, \dots, c_m$. We write $x_i \in c_j$ if variable x_i is present in clause c_j (either as a positive x_i or a negative $\neg x_i$ literal).

Each variable x_i is represented by a gadget which can be realized either clockwise or counterclockwise where clockwise realization corresponds to assigning TRUE value to variable x_i and counterclockwise realization corresponds to FALSE value of x_i. Each clause c_j is represented by several edges which can not be realized if all literals in c_j are false. For each clause c_j containing variable x_i the gadgets for x_i and c_j are connected by a pair of vertices (i.e. strings) $L_{c_j}^{x_i}$ and

$R_{c_j}^{x_i}$. The mutual position of $L_{c_j}^{x_i}$ and $R_{c_j}^{x_i}$ "transfers" the information about value of x_i to the clause.

Let G be an auxiliary incidence graph of variable-vertices and clause-vertices. Formally, $V_G = \{c_1, c_2, \dots, c_m\} \cup \{x_1, x_2, \dots, x_n\}$ and edges $E_G = \{\{x_i, c_j\} | x_i \in c_j, \forall i \in [n], \forall j \in [m]\}$. Let $D(G)$ be a straight line drawing of G with all vertices $x_1, x_2, \dots x_n$ on a common circle B in clockwise order and all edges and clause-vertices drawn inside the circle. We will use the notation $D(v)$ resp. $D(e)$ for the image of a vertex v resp. an edge e in the drawing D. Example of such graph and its drawing is given in Figure 3.2.

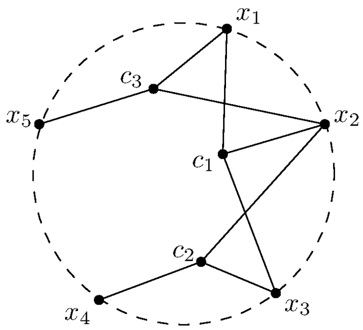

Figure 3.2: An auxiliar graph G of formula $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee x_4 \vee x_5)$ where clauses are named c_1, c_2 and c_3 from left to right.

Now we are ready to define graph H which has an outerstring representation if and only if the exact-3-CNF formula ϕ is satisfiable.

The idea of the construction is to replace each edge $x_i c_j$ in G by two vertices $L_{c_j}^{x_i}$, $R_{c_j}^{x_i}$ in H which can be realized by strings in some ϵ-neighborhood of $D(x_i c_j)$. Variable-vertices are replaced by *variable-gadgets* which ensure that the pairs of edges $L_{c_j}^{x_i}$, $R_{c_j}^{x_i}$ are for one variable realized in the same order. The variable-gadget uses construction of subset-cycle extension from Lemma 7. Clause-vertices are replaced by *clause-gadgets* that are realizable if and only if the corresponding clause is satisfied. Clause-gadgets consist of several incidences between edge-vertices $L_{c_j}^{x_i}$ resp. $R_{c_j}^{x_i}$ for variables x_i present in clause c_j similarly as in Kratochvíl [15].

Now we describe H in full detail:

We start with the set of vertices of H. Denote $\ell(x_i)$ number of clauses that contain variable x_i. Let $V(x_i)$ be the set of vertices of a *variable-gadget* of x_i. $V(x_i) = \left\{L_{c_1}^{x_i}, R_{c_1}^{x_i}, \dots L_{c_{\ell(x_i)}}^{x_i}, R_{c_{\ell(x_i)}}^{x_i}\right\}$[1] where each pair of vertices $L_{c_j}^{x_i}$, $R_{c_j}^{x_i}$ corresponds to some edge $x_i c_j$ in G. Moreover, we add auxiliary vertices $V'(x_i) = \{y(x_i)_1, y(x_i)_2, \dots y(x_i)_{5(2\ell(x_i)+1)}\}$ for each variable x_i. We define $V_H = \bigcup_{i=1}^n V(x_i) \cup V'(x_i)$.

Now we describe the edges of H and their purpose:

- For each variable x_i denote $x_i c_1, x_i c_2, \dots, x_i c_{\ell(x_i)}$ the outgoing edges in the clockwise order from the variable-vertex x_i in $D(G)$ and add to H edges of a subset-cycle extension which forces subset-ordering $L_{c_1}^{x_i} < R_{c_1}^{x_i} < L_{c_2}^{x_i} <$

[1]Here we slightly abuse the notation of indices of clauses but we don't want to further complicate the formal notation

$R^{x_i}_{c_2} < \cdots < R^{x_i}_{c_{\ell(x_i)}} < V \backslash \{V(x_i) \cup V'(x_i)\}$ as defined in the Lemma 7 using vertices $y(x_i)_1, y(x_i)_2, \ldots y(x_i)_{5(2\ell(x_i)+1)}$. We denote these edges E_1.

- We add edges between all vertices from the same variable-gadget. This ensures that even if the suset-cycle extension of the anchors in a variable-gadget will be realized in the reverse order than in G, the strings can leave the ϵ-neighborhood of the variable-gadget in the same order as in $D(G)$.

$$E_2 = \bigcup_{i=1}^{n} \{L^{x_i}_{c_j} L^{x_i}_{c_k}, R^{x_i}_{c_j} R^{x_i}_{c_k}, L^{x_i}_{c_j} R^{x_i}_{c_k}, R^{x_i}_{c_j} L^{x_i}_{c_k} | \forall j, k \in [\ell]; j \neq k\}$$

- For each crossing of edges $x_i c_j$, $x_m c_n$ in $D(G)$ we add edges $L^{x_i}_{c_j} L^{x_m}_{c_n}$, $R^{x_i}_{c_j} R^{x_m}_{c_n}$, $L^{x_i}_{c_j} R^{x_m}_{c_n}$, $R^{x_i}_{c_j} L^{x_m}_{c_n}$ in H. This allows us to create an outerstring representation of H similar to the drawing of G where strings representing $L^{x_i}_{c_j}$ and $R^{x_i}_{c_j}$ will be drawn next to each other in the ϵ-neighborhood of $D(x_i c_j)$.

$$E_3 = \{L^{x_i}_{c_j} L^{x_m}_{c_n}, R^{x_i}_{c_j} R^{x_m}_{c_n}, L^{x_i}_{c_j} R^{x_m}_{c_n}, R^{x_i}_{c_j} L^{x_m}_{c_n} || D(x_i c_j) \cap D(x_m c_n)| > 0\}$$

- For each clause-vertex c_j we add several edges which we will call a *clause-gadget* of c_j. Let x_a, x_b and x_c be variables used in c_j. Then we add edges $L^{x_a}_{c_j} L^{x_b}_{c_j}, L^{x_a}_{c_j} R^{x_b}_{c_j}, L^{x_a}_{c_j} L^{x_c}_{c_j}, R^{x_a}_{c_j} R^{x_b}_{c_j}, R^{x_a}_{c_j} L^{x_c}_{c_j}, R^{x_a}_{c_j} R^{x_c}_{c_j}, L^{x_b}_{c_j} L^{x_c}_{c_j}, L^{x_b}_{c_j} R^{x_c}_{c_j}, R^{x_b}_{c_j} R^{x_c}_{c_j}$. If ϕ contains negation of x_i, we swap $L^{x_i}_{c_j}$ and $R^{x_i}_{c_j}$ in the construction of edges so we treat these edge as if they had opposite orientation. We denote E_4 edges forced by all clause-gadgets.

3.3 Correctness

For the clause-gadget we need a lemma that a complement of the cycle of length six doesn't admit an outerstring representation in which the vertices are placed on the boundary in the same order as in the cycle.

Lemma 8 (Kratochvíl [14], Corollary of Claim 3)**.** *Let $G = (V_G, E_G)$ be a graph on six vertices $V_G = \{a, b, c, d, e, f\}$ with edges $E_G = \{ac, ad, ae, bd, be, bf, ce, cf, df\}$. There doesn't exist a constrained outerstring representation of G with vertices a, b, c, d, e, f placed in this order on the boundary.*

Lemma 9. *Let ϕ be an exact-3-CNF formula and graph H constructed as described above. Then if ϕ is satisfiable, H admits an outer-2-string representation and if ϕ is not satisfiable, H does not have an outerstring representation.*

Proof. If ϕ is satisfiable, we construct an outerstring representation of H. Let $x_1, x_2, \ldots, x_n$ be a satisfying assignment of ϕ. Let $D(G)$ be the drawing of G used to construct H. Because $D(G)$ is a straight line drawing, there exists some ϵ such that ϵ-neighborhoods of circle B, of all vertices and of all edges are disjoint except adjacent objects (e.g. circle B with variable-vertices and vertices with incident edges) and edges which cross with each other. We will use $N(O)$ for ϵ-neighborhood of an object O. From now on, $D(G)$ will be used only to define the place of variable-gadget in the ϵ-neighborhood of corresponding variable-vertex, clause-gadget in the ϵ-neighborhood of corresponding clause-vertex and strings connecting variable-gadgets and clause-gadgets in the ϵ-neighborhood of corresponding edges.

Firstly, we construct variable-gadgets. If $x_i = TRUE$ in the satisfying assignment, we place anchors of $L^{x_i}_{c_1}, R^{x_i}_{c_1}, L^{x_i}_{c_2}, \ldots, L^{x_i}_{c_{\ell(x_i)}}$ in the clockwise order on the boundary B in $N(x_i)$ and intersect strings as prescribed by E_2 in this neighborhood to get $L^{x_i}_{c_1}, R^{x_i}_{c_1}, L^{x_i}_{c_2}, \ldots, R^{x_i}_{c_{\ell(x_i)}}$ clockwise ordered on the border of $N(x_i)$. The intersection is shown in Figure 3.3.

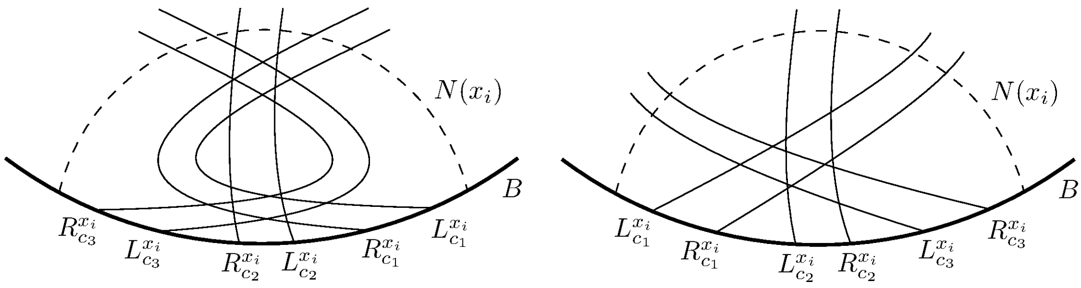

Figure 3.3: Intersections in $N(x_i)$ in case of clockwise (left) and counterclockwise (right) orientation of the anchors that ensures the clockwise orientation on the border of $N(x_i)$.

If $x_i = FALSE$ in the satisfying assignment, we place anchors of $L^{x_i}_{c_1}, R^{x_i}_{c_1}, L^{x_i}_{c_2}, \ldots, R^{x_i}_{c_{\ell(x_i)}}$ in the counterclockwise order in $N(x_i)$ and intersect strings as prescribed by E_2 in this neighborhood once to get $L^{x_i}_{c_1}, R^{x_i}_{c_1}, L^{x_i}_{c_2}, \ldots, R^{x_i}_{c_{\ell(x_i)}}$ clockwise ordered on the border of $N(x_i)$.

Now, we prolong all strings $L^{x_i}_{c_j}$ resp. $R^{x_i}_{c_j}$ to $N(c_j)$ through $N(x_i c_j)$. By this construction we create intersections corresponding to edges E_3. In $N(c_j)$ we intersect strings as shown in Figure 3.4. Because all clauses are satisfied, the three depicted cases cover all possibilities up to symmetry. This will ensure edges from E_4.

To finish the construction, we add all vertices and edges of subset-cycle extensions (i.e. vertices $\bigcup_{i=1}^{n} V'(x_i)$) in $N(B)$. Because the order of anchors of all vertices fulfills the ordering prescribed by subset-cycle extensions for each variable, it is possible to do that according to Lemma 7.

Observe that in our construction two strings intersect each other at most two times, so if ϕ is satisfiable, H has an outer-2-string representation.

If ϕ is not satisfiable, let us assume for contradiction that there exists an outerstring representation of H. From the construction of H, namely the addition of subset-cycle extensions for each variable, follows that for each variable x_i the anchors in variable-gadget of x_i are oriented either clockwise or counterclockwise. We define $x_i = TRUE$ if they are oriented clockwise and $x_i = FALSE$ if they are oriented counterclockwise. Because such assignment of all variables cannot satisfy ϕ, there must be a clause $c_j \in \phi$ which is not satisfied.

Let x_a, x_b and x_c be the variables in c_j. Then $L^{x_a}_{c_j}$, $R^{x_a}_{c_j}$, $L^{x_b}_{c_j}$, $R^{x_b}_{c_j}$, $L^{x_c}_{c_j}$, $R^{x_c}_{c_j}$ forms an outerstring representation of a complement of a cycle on six vertices where anchors of strings are in the order of the six cycle[2]. Such constrained

[2]Here we use notation assuming that all variables are in c_j as positive literals. Otherwise $L^{x_i}_{c_j}$ and $R^{x_j}_{c_j}$ would be realized in opposite orientation but also their neighbours in c_j would be swapped. So we would only need to rename the vertices but the resulting oriented subgraph would remain the same.

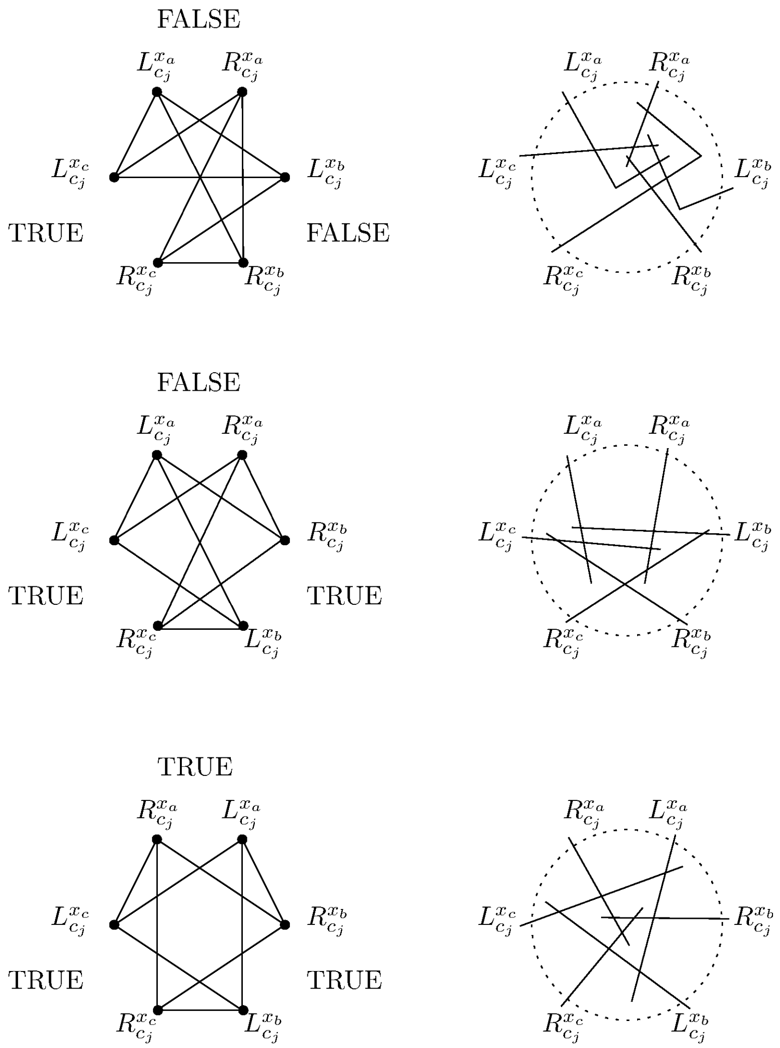

Figure 3.4: Possible intersection in $N(c_j)$ when at least one literal is realized clockwise (has TRUE value).

outerstring representation doesn't exist according to Lemma 8 (see Figure 3.5). □

Now we are ready to prove the main theorem.

Proof of Theorem 6. We will show a polynomial reduction of Exact-3-SAT to the recognition of outerstring graphs. Let ϕ be an exact-3-CNF formula. Graph H as described above can be constructed in polynomial time. According to Lemma 9 graph H has an outerstring representation if and only if ϕ is satisfiable and thus the problem of recognizing whether a graph admits an outer-2-string representation or not is at least as hard as Exact-3-SAT which is known to be NP-hard. □

From Theorem 6 follows that also finding an outerstring representation of an outerstring graph is NP-hard. Because we use this result to show that the maximum weight independent set algorithm by Keil et al. [13] needs an outerstring representation on its input and complexity of this algorithm depends on total number of piece-wise linear segments in the outerstring representation, we

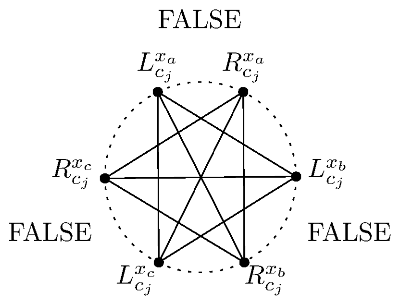

Figure 3.5: Unsatisfied clause corresponds to complement of six cycle. Its constrained outerstring representation doesn't exist.

will show that graph H as defined above admits a piece-wise linear outerstring representation with a few bends.

Proposition 10. *Graph H that can be used to find a satisfying assignment of an exact-3-CNF formula ϕ with n variables and m clauses admits a piece-wise linear outerstring representation with $60m + 5n$ linear segments.*

Proof. With a little bit careful placement we can realize all strings from subset-cycle extensions as single linear segments. There are $30m + 5n$ such strings in total.

Strings of vertices $L^{x_i}_{c_j}$ (resp. $R^{x_i}_{c_j}$) can be realized with one bend inside $N(x_i)$, one bend between $N(x_i)$ and $N(c_j)$ to arrive to the boundary of $N(c_j)$ in a similar way as depicted in Figure 3.4 (i.e. to form vertices of regular hexagon), one bound on the boundary of $N(c_j)$ and one possible bound inside $N(c_j)$ (in case of a clause with only one literal with TRUE value). That means 5 linear segments per string and $30m$ segments in total. $\square$

It would be possible to further reduce the total number of bends with more precise analysis but the important part of this proposition is that the total number of needed segments is linear with respect to the size of the formula ϕ.

Conclusion

In this thesis we studied the outer and grounded string representations of graphs. After mentioning previous results about classes we were interested in, we fill the remaining unresolved inclusions among them to provide a comprehensive overview of these classes and their mutual proper inclusions in Figure 2.1.

To prove the missing proper inclusions (resp. incomparability in case of Mpt) we used a powerful tool called Cycle Lemma (Lemma 1). Even though a similar lemma was previously known, we have generalized the lemma from outer-1-string graphs to outerstring graphs and thus greatly broaden the scale of problems for which it can be used. Because even the previously known version of Cycle Lemma is fairly new (published in 2016) and now it can be applied even to outerstring graphs, we believe that one of the most promising directions for future research would be to investigate problems that could be solved by this lemma.

In Section 3 we showed that the decision, whether a given graph admits an outer-2-string representation or it is not an outerstring graph at all, is NP-hard. This comes as another usage of the generalized Cycle Lemma. Unfortunately, we were unable in our reduction from Exact-3-SAT to encode the satisfiability of CNF formula ϕ into a graph G, that would admit an outer-1-string representation in case of ϕ being satisfiable. So the natural open question strengthening the result from Theorem 6 is:

Question. *Is the decision problem, whether a given graph G is an outer-1-string graph or it is not an outerstring graph, NP-hard?*

Bibliography

[1] A. R. Ahmed, F. De Luca, S. Devkota, A. Efrat, Md. I. Hossain, S. G. Kobourov, J. Li, S. A. Salma, and E. Welch. L-graphs and monotone L-graphs. *CoRR*, abs/1703.01544, 2017.

[2] A. Asinowski, E. Cohen, M. C. Golumbic, V. Limouzy, M. Lipshteyn, and M. Stern. Vertex intersection graphs of paths on a grid. *Journal of Graph Algorithms and Applications*, 16(2):129–150, 2012.

[3] S. Benzer. On the Topology of the Genetic Fine Structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11): 1607, 11 1959.

[4] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 12 1976.

[5] 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*, volume 10520 of *Lecture Notes in Comput. Sci.*, pages 153–166. Springer, Cham, 2017.

[6] D. Catanzaro, S. Chaplick, S. Felsner, B. V. Halldórsson, M. M. Halldórsson, T. Hixon, and J. Stacho. Max point-tolerance graphs. *Discrete Appl. Math.*, 216(part 1):84–97, 2017. ISSN 0166-218X. doi: 10.1016/j.dam.2015.08.019.

[7] J. Chalopin and D. Gonçalves. Every planar graph is the intersection graph of segments in the plane: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 631–638, New York, NY, USA, 2009. ACM.

[8] S. Chaplick and T. Ueckerdt. Planar Graphs as VPG-Graphs. pages 174–186. Springer, Berlin, Heidelberg, 2013.

[9] S. Chaplick, V. Jelínek, J. Kratochvíl, and T. Vyskočil. Bend-Bounded Path Intersection Graphs: Sausages, Noodles, and Waffles on a Grill. pages 274–285. Springer, Berlin, Heidelberg, 2012.

[10] C. P. Gabor, K. J. Supowit, and W. Hsu. Recognizing circle graphs in polynomial time. *Journal of the ACM*, 36(3):435–473, 7 1989. doi: 10.1145/65950.65951.

[11] D. Gonçalves, L. Isenmann, and C. Pennarun. Planar graphs as L-intersection or L-contact graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 172–184. Society for Industrial and Applied Mathematics, 2018.

[12] V. Jelínek and M. Töpfer. On grounded L-graphs and their relatives. 8 2018. URL `http://arxiv.org/abs/1808.04148`.

[13] J. M. Keil, J. S.B. Mitchell, D. Pradhan, and M. Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. *Computational Geometry*, 60(C):19–25, 1 2017. doi: 10.1016/j.comgeo.2016.05.001.

[14] J. Kratochvíl. String graphs. I. The number of critical nonstring graphs is infinite. *Journal of Combinatorial Theory, Series B*, 52(1):53–66, 5 1991. doi: 10.1016/0095-8956(91)90090-7.

[15] J. Kratochvíl. String graphs. II. recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 5 1991. doi: 10.1016/0095-8956(91)90091-W.

[16] J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B*, 53(1):1–4, 9 1991. doi: 10.1016/0095-8956(91)90050-T.

[17] J. Kratochvíl and J. Matoušek. Intersection Graphs of Segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 11 1994. doi: 10.1006/JCTB.1994.1071.

[18] J. Matoušek. Intersection graphs of segments and $\exists\{R\}$. *CoRR*, abs/1406.2636, 2014. URL `http://arxiv.org/abs/1406.2636`.

[19] S. McGuinness. On bounding the chromatic number of L-graphs. *Discrete Math.*, 154(1-3):179–187, 1996. doi: 10.1016/0012-365X(95)00316-O.

[20] M. Middendorf and F. Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Math.*, 108(1-3):365–372, 1992. doi: 10.1016/0012-365X(92)90688-C.

[21] A. Pawlik, J. Kozik, T. Krawczyk, M. Lasoń, P. Micek, W. T. Trotter, and B. Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *Journal of Combinatorial Theory, Series B*, 105:6–10, 3 2014. doi: 10.1016/J.JCTB.2013.11.001.

[22] A. Rok and B. Walczak. Outerstring graphs are χ-bounded. In *Computational geometry (SoCG'14)*, pages 136–143. ACM, New York, 2014.

[23] M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 9 2003. doi: 10.1016/S0022-0000(03)00045-X.

[24] A. M. S. Shrestha, S. Tayu, and S. Ueno. On two-directional orthogonal ray graphs. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1807–1810. IEEE, 5 2010. doi: 10.1109/ISCAS.2010.5537709.

[25] F. W. Sinden. Topology of Thin Film RC Circuits. *Bell System Technical Journal*, 45(9):1639–1662, 11 1966. doi: 10.1002/j.1538-7305.1966.tb01713.x.

[26] W. Wessel and R. Pöschel. On circle graphs. In *Graphs, hypergraphs and applications (Eyba, 1984)*, volume 73 of *Teubner-Texte Math.*, pages 207–210. Teubner, Leipzig, 1985.