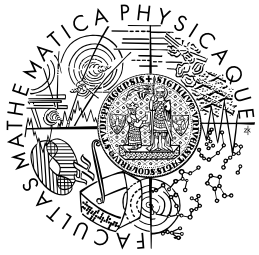

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Petr Chmel

Algorithmic aspects of intersection representations

Computer Science Institute of Charles University

Supervisor of the bachelor thesis: doc. RNDr. Vít Jelínek, Ph.D.

Study programme: Computer Science

Study branch: General Computer Science

Prague 2020

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

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 Kralupy n. Vlt., June 3, 2020 .....

Author's signature

I would like to thank my supervisor doc. RNDr. Vít Jelínek, Ph.D. for his guidance, patience, feedback and invaluable advice.

Title: Algorithmic aspects of intersection representations

Author: Petr Chmel

Institute: Computer Science Institute of Charles University

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

Abstract: As some problems are (NP-)hard to solve in the general case, a possible approach is to try to solve the problem on a restricted class of graphs. In the thesis, we focus on graphs induced by axis-aligned L-shapes, so-called L-graphs, and a similar class of axis-aligned L-shapes and ⅃-shapes, referred to as $\{\mathsf{L}, ⅃\}$-graphs, with two vertices sharing an edge if and only if their respective curves intersect. We show that recognizing both L-graphs and $\{\mathsf{L}, ⅃\}$-graphs is NP-complete. The second part of the thesis focuses on other typical decision problems on L-graphs and their relatives: finding the clique number, the independence number or a 3-coloring.

Keywords: intersection graph, L-graph, recognition, NP-completeness

Contents

Introduction

An intersection graph is a graph such that each vertex v of the graph can be assigned a set S_v and two sets S_u, S_v have a nonempty intersection if and only if the vertices u and v are connected by an edge. Usually, we consider classes of intersection graphs, which are restricted to a certain type of sets.

One of the possible uses of these restrictions is to make practically intractable problems (such as finding the maximum clique in a general graph) tractable by assuming additional constraints about the graph. There are classes of intersection graphs, for which at least some of the typical NP-complete problems can be solved in polynomial time, such as the intersection graphs of segments on a real line (so-called *interval graphs*), intersection graphs of subtrees in a tree (better known as the *chordal graphs* [1]) or the intersection graphs of chords in a circle (*circle graphs*).

As an example, assume that a university needs to know the least amount of virtual conference rooms needed to continue all the classes virtually during the lockdown so that an appropriate license is purchased. It is not difficult to see that this corresponds to a partition of the classes into conference rooms so that no two classes in the same conference room overlap with the minimum number of rooms. In terms of graph theory, this number is the chromatic number of an interval graph, where the vertices (classes) are represented on the real line as the time intervals during which the class takes place.

However, another issue may arise with these classes: in order to make use of the polynomial-time algorithms, we may need to know the intersection representations. For all three classes mentioned above, this is not an issue[1], as their representation can be constructed in polynomial time [2, 3, 4].

This is not true for other classes such as the intersection graphs of continuous nonnegative functions on closed intervals such that the values on their endpoints are zero (so-called *interval filament graphs*). This class admits a polynomial-time algorithm for finding the maximum weighted clique or the maximum weighted independent set [5] while being NP-complete to recognize [6].

Intersection graphs of curves in the plane (so-called *string graphs*) form another naturally occurring class of graphs. It is a folklore result that the class of string graphs is equivalent to the class of intersection graphs of paths in the grid, denoted as VPG. We may constrain the class further, as a VPG graph may contain a vertex represented by a large number of bends. As such, the class B_k-VPG is defined to be a subclass of VPG, where all of the paths have at most k bends. For all of these classes, recognition is NP-complete. In the case of string graphs, this was a remarkable long-standing open problem, as NP-hardness [7] was proved by Kratochvíl more than ten years before Schaefer, Sedgwick and Štefankovič showed that recognizing string graphs is in NP [8]. For B_k-VPG graphs, the case of $k = 0$ was shown to be NP-complete by Kratochvíl and Matoušek [9] and the remaining case with $k \geq 1$ was shown to be NP-complete by Chaplick et al.[10].

In this thesis, we focus on three subclasses of B_1-VPG: L-graphs, $\{\mathsf{L}, \mathsf{J}\}$-graphs

[1] Chordal graphs are a slight exception, as most of the algorithms for this class use the fact that every chordal graph has a perfect elimination ordering.

and grounded L-graphs. The first two classes are restricted based on permitted shapes – in the first case, only L-shapes are allowed, in the second case both L-shapes and ⅃-shapes are allowed. The third class is another restriction of L-graphs. In particular, in every grounded L-graph, there exists a line such that the highest points all L-shapes touch the line.

In chapter 1, we build the foundation for the following chapters and mention more related works. Chapter 2 contains the proof that recognizing both L-graphs and $\{\mathsf{L}, ⅃\}$-graphs is NP-complete, which answers an open problem posed by Felsner et al. [11]. Next, we study other decision problems on L-graphs and grounded L-graphs in chapter 3.

Preliminaries

We will use standard notation as is usual in the introductory courses on discrete mathematics and computational complexity. In particular, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We now introduce some more advanced concepts which will be used extensively in the thesis.

An **intersection graph** of a family of sets $\mathcal{M}$ is a graph such that for every set $M \in \mathcal{M}$, the graph contains a vertex v_M corresponding to the set and there exists en edge between two vertices $v_M, v_N : v_M \neq v_N$ if and only if $M \cap N \neq \emptyset$.

A **representation** R of a graph $G = (V, E)$ is a family of piecewise linear curves $R = \{R(v) : v \in V\}$ such that $\forall u, v \in V : R(v) \cap R(u) \neq \emptyset \Leftrightarrow \{u, v\} \in E$.

A representation is **proper** if every curve is simple, there are finitely many intersection points and finitely many bends, and in every point of the plane, at most two curves intersect and every such intersection is a crossing.

A **string graph** is an intersection graph of curves in the plane.

An **$\mathbf{B}_k$-VPG graph** is an intersection graph of paths in a grid with at most k bends.

An **L-shape** is the union of a horizontal line segment and a vertical line segment such that the only common point of the two line segments is the lowest point of the vertical line segment and the leftmost point of the horizontal line segment.

An **L-representation** is a representation such that every curve is an L-shape.

An **L-graph** is an intersection graph of L-shapes with a proper L-representation.

An **⅃-shape** is the union of a horizontal line segment and a vertical line segment such that the only common point of the two line segments is the lowest point of the vertical line segment and the rightmost point of the horizontal line segment.

An **$\{\mathsf{L}, ⅃\}$-representation** is a representation such that every curve is an L-shape or an ⅃-shape.

An **$\{\mathsf{L}, ⅃\}$-graph** is an intersection graph of L-shapes and ⅃-shapes with a proper $\{\mathsf{L}, ⅃\}$-representation.

A **grounded L-graph** is an intersection graph of L-shapes with a proper L-representation such that the topmost points of the L-shapes lie on a single line.

A **circle graph** is an intersection graph of chords in a circle. Equivalently, it is an intersection graph of L-shapes in a circle such that both endpoints of the L-shape touch the circle boundary.

1. Related works

This chapter contains previously known relations between graph classes and hardness results in more detail.

We mainly focus on three related classes: grounded L-graphs, L-graphs, and $\{\mathsf{L}, \lrcorner\}$-graphs. These are quite similar and for many decision problems, these graph classes lie near the boundary between classes with polynomial-time algorithms and classes for which the problem is NP-complete.

1.1 Investigated decision problems

The main problem which occurs naturally with every graph class is the recognition of the class. We formulate the problem for L-graphs and $\{\mathsf{L}, \lrcorner\}$-graphs. The case of grounded L-graphs is still open.

Definition 1 (L-graph recognition)**.**

L-graph recognition
Instance: A graph $G = (V, E)$.
Question: Does G have a proper L-representation (i.e., is it an L-graph)?

Definition 2 ($\{\mathsf{L}, \lrcorner\}$-graph recognition)**.**

$\{\mathsf{L}, \lrcorner\}$-graph recognition
Instance: A graph $G = (V, E)$.
Question: Is G an $\{\mathsf{L}, \lrcorner\}$-graph?

However, we will mostly deal with restricted decision problems which are NP-complete in a general setting. The three main problems are Clique, Independent Set and 3-Coloring. We will also consider generalisations of 3-Coloring such as k-Coloring and Coloring.

Definition 3 (Clique)**.**

Clique
Instance: A graph $G = (V, E)$, an integer $k \in \mathbb{N}$.
Question: Does there exist a clique (a complete subgraph) in G of size k?

Definition 4 (Independent Set)**.**

Independent Set
Instance: A graph $G = (V, E)$, an integer $k \in \mathbb{N}$.
Question: Does G have an independent set of size k as an induced subgraph?

Definition 5 (3-Coloring)**.**

3-Coloring
Instance: A graph $G = (V, E)$.
Question: Is G 3-colorable?

Definition 6 (k-Coloring)**.**

k-Coloring
Instance: A graph $G = (V, E)$.
Question: Is G k-colorable?

Definition 7 (Coloring).

Coloring
Instance: A graph $G = (V, E)$, an integer $k \in \mathbb{N}$.
Question: Is G k-colorable?

There is a subtle but significant difference between k-Coloring and Coloring. In k-Coloring, k is not a part of the input and is therefore a constant, whereas in Coloring, k is a part of the input. In particular, we need $\log k$ bits to represent k.

If we could formulate an algorithm for finding a k-coloring with time complexity $\mathcal{O}(n^k)$, it would be polynomial for k-Coloring, but it would not be polynomial for Coloring. The time complexity is double exponential in the size of the input, as the size of k on input is only $\log k$.

1.2 L-graphs and similar classes

The first chain of observations is simple: every L-graph is an $\{\mathsf{L}, \mathsf{J}\}$-graph and every $\{\mathsf{L}, \mathsf{J}\}$-graph is a B_1-VPG graph. At the same time, it is obvious that every grounded L-graph is an L-graph.

The fact that every B_0-VPG graph is an L-graph is not immediately obvious, given that we only consider proper L-graphs. To show this, we first need to remark that every interval graph is a grounded L-graph. This follows easily, as we may represent the intervals as L-shapes with the same length of the horizontal line segment such that in every pair of L-shapes, the L-shape which corresponds to the interval whose left endpoint is more to the left has its bend higher. In case two or more intervals share the same left endpoint, we may perturb them so that the endpoints are unique. In the case of vertical lines, we reflect the line across the axis of symmetry of the first and the third quadrant, so that the order of endpoints from left to right corresponds to the order of the original endpoints from bottom to top. This is necessary for the construction in the next paragraph.

Now, given a B_0-VPG graph, every line which contains at least one line segment is an interval graph and, therefore, can be represented as an arbitrarily thin grounded L-graph. In the case of vertical lines, we might change the "grounding" of the L-shapes so that the rightmost points of the L-shapes lie on the same line. This can be easily achieved by reflecting the grounded L-representation with the axis of reflection being the axis of symmetry of the first and the third quadrant. Then we add the reflected grounded L-graphs on the position of the line they represent, which creates an L-representation.

Another useful inclusion is that every circle graph is a grounded L-graph. This follows from an equivalent characterization of circle graphs, which is that every circle graph is an intersection graph of L-shapes drawn inside a circle with both endpoints touching the circle's boundary [12]. We may extend the L-shapes so that they touch a tangent of the circle which is perpendicular to the vertical line segments of the L-shapes. Note that we cannot add any more intersections by the extension.

There is a history of reducing the necessary number of bends for planar graphs as B_k-VPG. First, three bends were shown to be sufficient and conjectured to be necessary by Asinowski et al. [12]. This conjecture was disproved by Chaplick and

Ueckerdt who showed that only two bends are sufficient [13]. This was strengthened by Biedl and Derka who proved that planar graphs can be represented with two bends and every pair of vertices sharing at most a single intersection point [14]. The latest result by Gonçalves, Isenmann, and Pennarun shows that every planar graph is an L-graph [15].

It is also worth mentioning that grounded L-graphs have an alternative characterization. Jelínek and Töpfer showed that every graph is a grounded L-graph if and only if it admits a vertex order with forbidden patterns of size 4 [16].

By way of context, we mention a graph class similar to VPG – the class of edge intersection graphs of paths on a rectilinear grid, where two vertices are adjacent if and only if their corresponding paths share an edge. This class is usually written as EPG. In an analogy to B_k-VPG, B_k-EPG is the class of edge intersection graphs with paths which have at most k bends.

1.3 Recognizing the classes

The main results of the thesis are NP-completeness of Recognition of L-graphs and Recognition of $\{L, J\}$-graphs. This resolves an open problem posed by Felsner et al. [11] as the case of B_k-VPG for $k \geq 1$ and $\{L, 7\}$-graphs was shown to be NP-complete by Chaplick et al [10]. The method used in the proof is similar and uses the same technical tool which is formulated in Section 2.

The complexity of recognizing grounded L-graphs is still open despite the characterization via forbidden patterns. While it is possible to recognize all combinations of 3-vertex forbidden patterns in polynomial time, this is not possible in general for 4-vertex patterns [17].

The analogous EPG graphs are studied similarly and the results are similar as well, yet the techniques used differ significantly. Heldt, Knauer, and Ueckerdt [18] proved that the recognition of B_1-EPG is NP-complete and Cameron, Chaplick, and Hoàng [19] showed the same holds for all natural subclasses of the shapes, i.e., $\{L\}, \{L, J\}, \{L, 7\}$ and $\{L, \Gamma, 7\}$. This was further extended for B_2-EPG and its natural subclasses by Pergel and Rzążewski [20].

1.4 Other problems on the classes

1.4.1 Clique

Clique is a problem, which is already known to be in P for $\{L, J\}$-graphs and NP-complete for $\{L, 7\}$-graphs, both proved by Middendorf and Pfeiffer [21]. Their algorithm based on dynamic programming has time complexity $\mathcal{O}(n^4)$ for $\{L, J\}$-graphs and $\mathcal{O}(n^3)$ for grounded $\{L, J\}$-graphs, whereas the algorithm discussed in the thesis has better time complexity in exchange for restricting the graph class.

1.4.2 Independent Set

Independent Set is also known to be NP-complete for B_0-VPG [22] and therefore is also NP-complete on L-graphs. At the same time, a recent result of Bose

et al. yields an $(4 \cdot \log \mathrm{OPT})$-approximation algorithm for L-graphs where OPT is the size of an optimal solution [23].

We provide another proof of NP-completeness of Independent Set on L-graphs, which is based on a reduction from a special case of SAT, in particular the Planar Monotone Rectilinear 3-SAT which was shown to be NP-complete by de Berg and Khosravi [24].

1.4.3 Coloring

The situation with deciding the chromatic number of a graph is a little more complicated. Both B_0-VPG and circle graphs do not offer any help with Coloring as the problem is still NP-complete even with the restriction [11, 25]. The case of 3-Coloring is open for grounded L-graphs. However, from the fact that every planar graph is an L-graph and that 3-Coloring is NP-complete when restricted to planar graphs with degree at most 4 [26], we see that 3-Coloring is NP-complete on L-graphs as well. The same does not apply for triangle-free[1] planar graphs as Grötzsch's theorem [27] shows that all triangle-free planar graphs are 3-colorable and therefore 3-Coloring is in P when restricted to triangle-free planar graphs.

We prove NP-completeness of 3-Coloring on L-graphs and triangle-free L-graphs with a given representation, also reducing from Planar Monotone Rectilinear 3-SAT. We also discuss possible approximation algorithms for Coloring grounded L-graphs.

[1] A graph is triangle-free if it does not contain a triangle (K_3) as a subgraph.

2. Recognizing the classes

This chapter focuses on the main results of the thesis - the NP-completeness of L-graph recognition and $\{\mathsf{L}, \mathsf{\lrcorner}\}$-graph recognition.

Theorem 1. L-graph recognition *is NP-complete.*

Theorem 2. $\{\mathsf{L}, \mathsf{\lrcorner}\}$-graph recognition *is NP-complete.*

2.1 Building blocks

To prove the theorems, we will use the same approach Chaplick et al. [10] used for proving NP-completeness of recognizing B_k-VPG graphs for $k \geq 1$. Their approach extensively uses terms related to geometric representations which are defined below.

Definition 8 (Intersection point)**.** An *intersection point* of a representation R is a point in the plane which belong to two (or possibly more) distinct curves in R. We will use $\mathrm{In}(R)$ to denote the set of all intersection points in R.

Definition 9 (Order-preserving mapping)**.** Let R, R' be two representations of G, where R is proper (R' does not have to be proper). Let φ be a mapping $\varphi\colon \mathrm{In}(R) \to \mathrm{In}(R')$.

We say that φ is *order-preserving* if it is injective and for every $v \in V$, if $p_1, \ldots, p_k$ are all distinct intersection points on $R(v)$, then $\varphi(p_1), \ldots, \varphi(p_k)$ all belong to $R'(v)$ and they appear on $R'(v)$ in the same relative order as the points $p_1, \ldots, p_k$ on $R(v)$.

An extremely useful tool is the *Noodle-Forcing Lemma* by Chaplick et al.

Lemma 3 (Noodle-Forcing Lemma, Chaplick et al., 2012 [10])**.** *Let $G = (V, E)$ be a graph with a proper representation $R = \{R_v : v \in V\}$. Then there exists a graph $G' = (V', E')$ containing G as an induced subgraph, which has a proper representation $R' = \{R'(v) : v \in V'\}$ such that $R(v) = R'(v)$ for all $v \in V$ and $R'(w)$ is a vertical or a horizontal segment for $w \in V' \setminus V$.*

Moreover, for any $\varepsilon > 0$ any (not necessarily proper) representation of G' can be transformed by a homeomorphism of the plane and a circular inversion into a representation $R^\varepsilon = \{R^\varepsilon(v) : v \in V'\}$ with these properties:

1. *for every vertex $v \in V$, the curve $R^\varepsilon(v)$ is contained in the $\varepsilon-$neighborhood of $R(v)$ and $R(v)$ is contained in the $\varepsilon-$neighborhood of $R^\varepsilon(v)$.*

2. *there is an order-preserving mapping $\phi : In(R) \to In(R^\varepsilon)$ with the additional property that for every $p \in In(R)$, the point $\phi(p)$ coincides with the point p.*

Sketch of the proof. We are given a proper representation R of G. First, we define *special* points of R, which are endpoints of the curves, bends of the curves and intersection points of the curves.

We first construct an auxiliary grid graph H so that its drawing overlays the representation with the following properties:

P1 The edges of H are drawn as vertical and horizontal segments, and every internal face of H is a rectangle. Moreover, the outer face of H is not intersected by any curve of R.

P2 No curve of R passes through a vertex of H and no edge of H passes through a special point of R.

P3 Every face of H contains at most one special point of R and no two faces containing a special point are adjacent.

P4 Every edge of H is intersected at most once by the curves of R.

P5 Every face of H is intersected by at most two curves of R, and if a face f is intersected by two curves of R, then the two curves intersect inside the face f.

P6 Every curve of R intersects the boundary of a face of H at most twice.

This is done in four steps, starting with a grid fine enough so that properties P1-P3 hold. Adding the next three properties is done by splitting the faces.

The grid of H is then converted into the graph G' induced by its representation R'. For any vertex v of the grid H, we create two vertices $S_1(v), S_2(v)$ joined by an edge. Then for every edge $e \in E(P)$ incident with v, we create another vertex $S(v, e)$, which is joined to $S_1(v)$ if the edge is horizontal and to $S_2(v)$ if e is vertical. We also take every edge $e = \{u, v\} \in E(H)$ and add the vertex $S(e)$, which is adjacent precisely to vertices $S(u, e)$ and $S(v, e)$. Finally, we add vertices of G with the same representation and add edges between the vertices and edge-vertices $S(e)$ of the grid which their curves cross. A part of the construction is sketched with L-shapes in Figure 2.7.

These six properties are then enough to prove the rest of the statement. □

The underlying reduction used in proving the Theorems 1 and 2 also uses a result from Kratochvíl [28], who showed that the recognition of grid intersection graphs (sometimes referred to as PURE-2-DIR graphs) is NP-complete. We sketch the reduction used in proving the NP-completeness of Grid intersection graph recognition as it will be used later for our proof of Theorems 1 and 2. The reduction is based on the NP-completeness of 4-bounded Planar 3-connected 3-SAT, which we write as 4P3C3SAT for short.

Definition 10 (4P3C3SAT)**.**

4P3C3SAT
Instance: A formula φ with a set of clauses C over a set of variables X satisfying the following: 1. every clause contains exactly three distinct variables 2. every variable occurs in at most 4 clauses 3. the bipartite graph $G_\varphi = (X \cup C, \{\{x, c\} : x \in c \in C \vee (\neg x) \in c \in C\})$ is planar and vertex 3-connected. **Question:** If φ satisfiable?

Theorem 4 (Kratochvíl, 1994 [28]). *The decision problem* 4P3C3SAT *is NP-complete.*

Definition 11 (Grid intersection graph). A *grid intersection graph* is an intersection graph whose vertices are represented by line segments on a grid in the plane with the additional property that every two collinear line segments do not intersect.

Definition 12 (Grid intersection graph recognition).

Grid intersection graph recognition
Instance: A graph $G = (V, E)$.
Question: Is G a grid intersection graph?

Theorem 5 (Kratochvíl, 1994 [28]). *The decision problem* Grid intersection graph recognition *is NP-complete.*

Sketch of the construction. Given an instance φ of 4P3C3SAT, we construct G'_φ in the following steps:

1. We fix an embedding D_φ of $G_\varphi = (X \cup C, \{\{x, c\} : x \in c \in C \vee (\neg x) \in c \in C\})$ in the grid such that the edges are piecewise linear and lie on the grid lines, no two edges cross and no vertex lies on an edge.

2. For every edge $e = \{x, c\} \in E(G_\varphi)$, we use $d(e)$ to denote the number of linear pieces in the drawing of e. Taking $L(e), R(e)$ to be two distinct paths on $d(e)$ vertices $l_1(e), l_2(e), \ldots, l_{d(e)}(e), r_1(e), r_2(e), \ldots, r_{d(e)}(e)$, we set $G_e = L(e) \cup R(e)$.

3. We consider graph F as in Figure 2.1, called the *frame*. The role of the frame is to restrict the construction to a bounded convex region as in every segment representation of F, there is exactly one region Ω which is bounded by the four segments a, b, c, d. Moreover, the region is convex. The segments denoted by a, b, c, d in the figure correspond to the vertices a, b, c, d in the following step.

4. We also consider an uppermost, a rightmost, a bottommost and a leftmost linear segment of D_φ. Let the segments be the $i(u)$th, $i(r)$th, $i(b)$th and $i(l)$th linear piece of an edge e_u, e_r, e_b, e_l. Then, we take $G_l = (V_l, E_l)$ with $V_l = \{t(u), t(r), t(b), t(l), a, b, c, d, l_{i(u)}(e_u), l_{i(r)}(e_r), l_{i(b)}(e_b), l_{i(l)}(e_l), r_{i(u)}(e_u), r_{i(r)}(e_r), r_{i(b)}(e_b), r_{i(l)}(e_l)\}$ and $E_l = \{\{a, t(u)\}, \{b, t(r)\}, \{c, t(b)\}, \{d, t(l)\}, \{t(u), l_{i(u)}(e_u)\}, \{t(r), l_{i(r)}(e_r)\}, \{t(b), l_{i(b)}(e_b)\}, \{t(l), l_{i(l)}(e_l)\}\}, \{t(u), r_{i(u)}(e_u)\}, \{t(r), r_{i(r)}(e_r)\}, \{t(b), r_{i(b)}(e_b)\}, \{t(l), r_{i(l)}(e_l)\}$.

   This graph attaches the gadgets described below to the frame.

5. For every variable x, we construct a variable gadget G_x as follows. If x appears in four clauses, we number the clauses in the clockwise order on the edges leaving the vertex x in D_φ. Then, we set $A_i = l_1(\{x, c_i\}), B_i = r_1(\{x, c_i\})$ if $x \in c_i$ and $A_i = r_1(\{x, c_i\}), B_i = l_1(\{x, c_i\})$ if $\neg x \in c_i$. The gadget itself is depicted in Figures 2.2 and 2.3 in two versions: one for a variable which is in four clauses and another for a variable which is in three clauses.

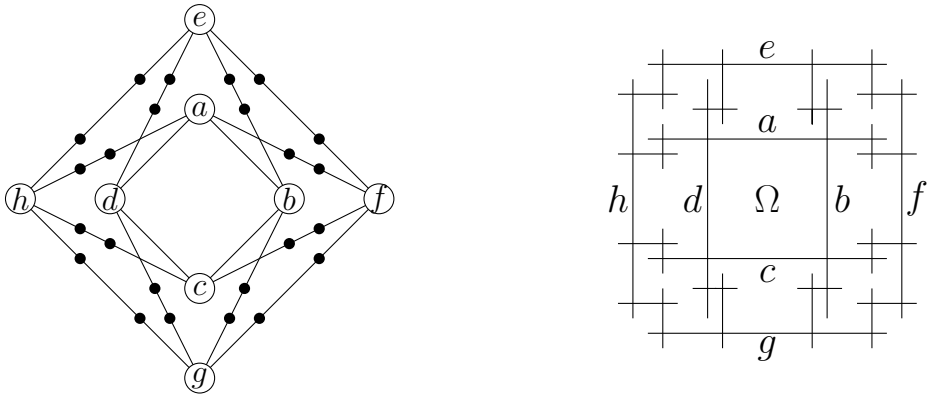

Figure 2.1: The frame F with its PURE-2-DIR representation

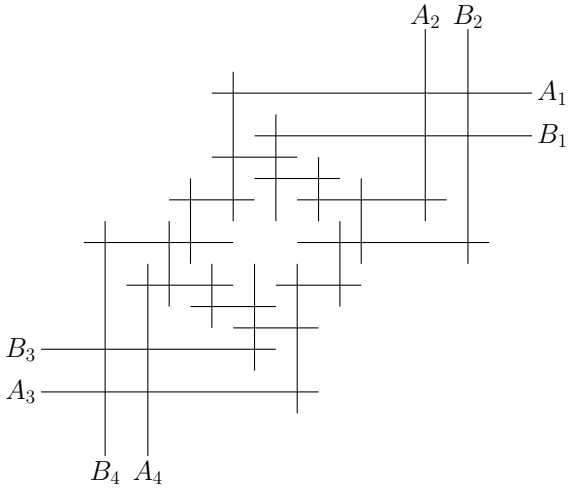

Figure 2.2: The variable gadget with four occurrences

6. For each clause c, we construct a clause gadget G_c in the following way. Let x_1, x_2, x_3 be the variables occurring in c numbered in the clockwise order as the edges $x_i c$ are around c in the drawing D_φ. The gadget is depicted in Figure 2.4. It is based on the frame, so that all intersection points of vertices $u_1, u_2, u_3, v_1, v_2, v_2, w$ lie in the bounded convex region.

7. We set $G'_\varphi = (V, E)$, where $V = V(F) \cup \bigcup_{e \in E(G_\varphi)} V(G_e) \cup V_l \cup \bigcup_{x \in X} V(G_x) \cup \bigcup_{c \in C} V(G_c)$ and $E = E(F) \cup \bigcup_{e \in E(G_\varphi)} E(G_e) \cup E_l \cup \bigcup_{x \in X} E(G_x) \cup \bigcup_{c \in C} E(G_c)$.

The steps 3 and 4 anchor the representation in a bounded convex region. The gadgets then simulate the true/false assignment by the order of the paths l and r.

This means that in PURE-2-DIR, the clause gadget cannot be represented, if the vertices $r_{d(\{x_1,c\})}(\{x_1, c\}), l_{d(\{x_1,c\})}(\{x_1, c\}), r_{d(\{x_2,c\})}(\{x_2, c\}), l_{d(\{x_2,c\})}(\{x_2, c\})$, $r_{d(\{x_3,c\})}(\{x_3, c\}), l_{d(\{x_3,c\})}(\{x_3, c\})$ intersect the gadget in this clockwise order. This order then also means that the vertices $u_1, v_1, u_2, v_2, u_3, v_3$ intersect the vertices a, b, c, d of the clause gadget's frame in the same order and it is impossible to represent the gadget then.

As it can be shown that the vertices u_1, v_2 and v_1, v_2 intersect inside the region Ω and the same applies for v_3, u_3 intersecting u_2 and v_3 intersecting v_2, the vertex w can only be drawn inside the region Ω as well. However, with the order of

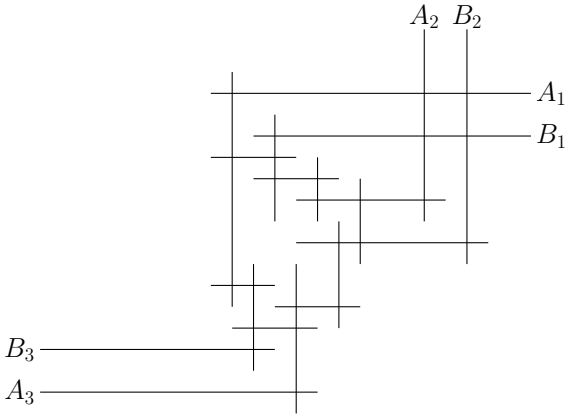

Figure 2.3: The variable gadget with three occurrences

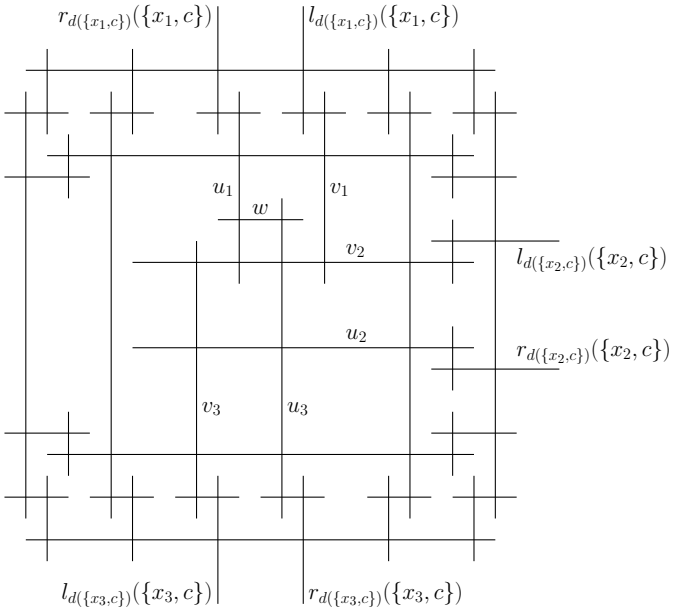

Figure 2.4: The clause gadget

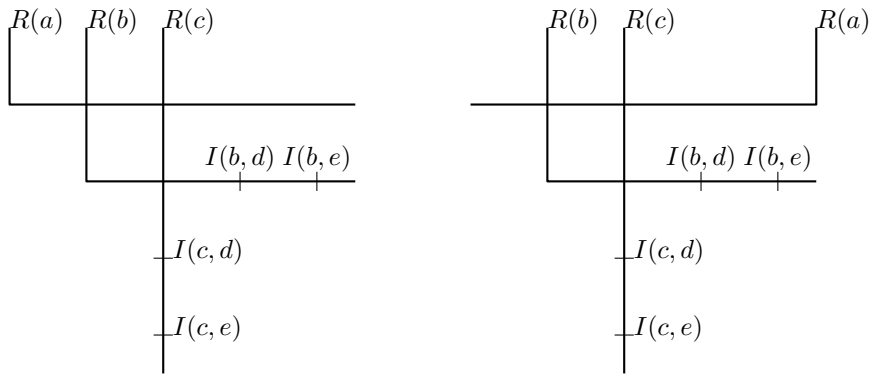

Figure 2.5: The two possible situations in the proof of Lemma 6

vertices $u_1, v_1, u_2, v_2, u_3, v_3$, the segments v_1, v_2, v_3 divide Ω into two regions with boundaries $dav_1v_2v_3c$ and $bcv_3v_2v_1a$ and u_1 must be only in the first region while u_3 can only be in the second region and hence we cannot add the line segment corresponding to vertex w. □

Next, we prove a lemma that enables us to use the Noodle-Forcing Lemma to force the bend on a vertex.

Lemma 6. *For a proper $\{\mathsf{L}, \mathsf{J}\}$-representation R of $K_5, V(K_5) = \{a, b, c, d, e\}$, if the intersection points on all five vertices are located in relative alphabetical order, then the vertex c cannot be represented just as a horizontal or a vertical line segment.*

Proof. We prove the lemma by contradiction. To simplify notation, we use $R(x)$ to denote the representation of the vertex x and $I(x, y)$ denotes the intersection point of $R(x)$ and $R(y)$.

First, let c be represented as a horizontal line segment $R(c)$. This means that all bends of vertices a, b, d, e must be below $R(c)$, otherwise they could not intersect c. We immediately get a contradiction, as all intersection points on $R(a)$ must be below the intersection point with $R(c)$, and therefore the order of intersection points on $R(a)$ starts or ends with c, which cannot happen.

Next, let c be represented as a vertical line segment $R(c)$. As the situations are symmetric, we may assume without loss of generality that the order of intersection points on $R(c)$ is a, b, d, e with a being the highest and e the lowest. In the same vein, we have a similar symmetry with the position of the intersection point $I(a, b)$ of $R(a)$ and $R(b)$. If $I(a, b)$ was to the right of the vertical line segment $R(c)$, we may use reflection with $R(c)$ as the axis of reflection and $I(a, b)$ would then be to the left of $R(c)$. From now on, we assume $I(a, b)$ to be to the left of $R(c)$.

Given this, we know that b has to be represented as an L-shape as it could not intersect both $R(a)$ and $R(c)$ otherwise. The two possible situations are depicted in Figure 2.5. Therefore, it is necessary that vertices d and e are represented as J-shapes so that they can intersect both vertices b and c in the prescribed order.

Now, we observe that the intersection points $I(b, d)$ and $I(b, e)$ must lie on $R(b)$ in this order, therefore the x-coordinate of the bend of $R(d)$ is less than the x-coordinate of the bend of $R(e)$. At the same time, from the order of intersection points on $R(c)$, we know that the y-coordinate of the bend of $R(d)$ must be higher than the y-coordinate of the bend of $R(e)$. From these two constraints, we see that

$R(d)$ and $R(e)$ cannot intersect, which is the contradiction with the expectation that the represented graph is complete. □

2.2 Proofs of NP-completeness

Proof of Theorem 1. It is not difficult to see that L-GRAPH RECOGNITION is in NP. We can imagine that the L-shapes are lying on a grid, therefore the polynomial certificate for any L-representation would consist of four integers for each L-shape: the two coordinates of the intersection of the horizontal and the vertical line segments and the two lengths of the segments. The coordinates and lengths can surely be bounded by n. Verifying the intersections and nonintersections then surely takes polynomial time - there are $\Theta(n^2)$ pairs of L-shapes and verifying whether a single pair intersects takes constant time given the certificate described above.

Next, we show the decision problem is NP-hard.

We use a construction which is very similar to the one used in the proof of the NP-completeness of recognition of B_k-VPG graphs. The terminology is also quite similar, as we will construct a *pin* and a *clothespin*, from which we will be able to build a polynomial reduction from the recognition of grid intersection graphs.

In order to construct the pin, we construct an auxiliary graph H first. First, we take a K_5 with its vertices a, b, c, d, e and create its L-representation R so that the intersection points on all five vertices are located in relative alphabetical order. Then, we overlay the representation with a planar grid graph P according to Figure 2.6. This grid satisfies all six properties P1-P6 from the proof of the Noodle-Forcing Lemma and therefore Lemma 3 applies. From this, we construct the graph H induced by its representation R' as in the Noodle-Forcing Lemma: for any vertex v of the grid, we create two vertices $S_1(v), S_2(v)$ joined by an edge. Then for every edge $e \in E(P)$ incident with v, we create another vertex $S(v, e)$, which is joined to $S_1(v)$ if the edge is horizontal and to $S_2(v)$ if e is vertical. We also take every edge $e = \{u, v\} \in E(H)$ and add the vertex $S(e)$, which is adjacent precisely to vertices $S(u, e)$ and $S(v, e)$. Finally, we add the five vertices a, b, c, d, e from the K_5 and add edges between the vertices and edge-vertices $S(e)$ of the grid which their L-shapes cross. A part of the construction is depicted in Figure 2.7.

We will write this as the pin $P(c)$, where c is the middle vertex of the K_5 representation. As the grid (via the Noodle-Forcing Lemma) forces the order of intersections, the Lemma 6 guarantees that c has a bend. Also, we may create a representation of $P(c)$ such that the last segment of $R'(c)$ may be extended arbitrarily far, as shown in the schematic Figure 2.8. We will refer to the last segment as the *tip* of $P(c)$, we will also say that c is an *exposed* vertex, and the cell of the grid which is "inverted" will be denoted by α. Note that in this way, we are able to create a pin with a horizontal tip and a pin with a vertical tip.

This, however, is not enough to make use of Kratochvíl's construction. It may still happen that the intersection of two exposed vertices could happen in a cell of the grid different from α – one pin could be fully contained in a cell of the other pin.

In order to tackle the issue, we will construct the clothespin. We start with a K_4, whose edges are all subdivided once. Then, we take two faces of the drawing

and add vertices m_1, m_2 connected by paths of length two to the three original vertices of K_4 from its respective face. Next, for each of m_1, m_2, we add a path of length two connecting it with the outside of the pin $P(v_1)$ or $P(v_2)$ respectively. These pins are also connected to a vertex which was subdividing the edge of the original K_4. Lastly, the tips of the two pins are connected to the tip of a third pin. It may not happen that the pin vertices intersect anywhere else than in their pins, as in that case, they would have to intersect at least one other vertex. The construction is depicted in Figure 2.9.

The clothespin $CP(v)$ then consists of three pins $P(v_1), P(v_2), P(v_3)$ so that precisely the pairs v_1, v_3 and v_2, v_3 form an edge.

Now, we may use our clothespin construction in the original construction by Kratochvíl. Given a formula φ, we construct G^{L}_{φ} by taking the construction G'_{φ} from the proof of Theorem 5. Then we replace every vertex $v \in V(G'_{\varphi})$ by the clothespin $CP(v)$. If two vertices u, v share an edge, then we add four new edges $u_i v_j : i, j \in \{1, 2\}$, where u_i, v_j are the exposed vertices.

It remains to verify that φ is satisfiable if and only if G^{L}_{φ} has a proper L-representation.

Given a satisfiable φ and its satisfying assignment a, we create a grid intersection graph in the same way as Kratochvíl [28] and replace every vertex with a clothespin with the bodies of the pins small enough so that they do not intersect anything else, which is a proper L-representation of G^{L}_{φ}.

On the converse, if G^{L}_{φ} has a proper L-representation, then the tips of $R'(u_1)$ for $u \in V(G'_{\varphi})$ form a GIG representation of G'_{φ}, and φ is satisfiable. □

The proof of Theorem 2 is similar, only similar arguments are presented for $\{\mathsf{L}, \mathsf{J}\}$-representations instead and there is another possibility of representing the horizontal pin, with the "head" of the pin being to the right of the tip.

Another observation is that given a graph class $\mathcal{C}$ such that L-graphs are a subclass of $\mathcal{C}$ and $\mathcal{C}$ is a subclass of $\{\mathsf{L}, \mathsf{J}\}$-graphs, recognition of $\mathcal{C}$ is NP-complete. We showed that if φ is satisfiable, then G^{L}_{φ} is an L-graph (and therefore $G^{\mathsf{L}}_{\varphi} \in \mathcal{C}$), while if φ is not satisfiable, then G^{L}_{φ} is not an $\{\mathsf{L}, \mathsf{J}\}$-graph (and $G^{\mathsf{L}}_{\varphi} \notin \mathcal{C}$).

Much like Chaplick et al. showed for all B_k-VPG graphs that the recognition of B_k-VPG graphs when given a B_{k+1}-VPG representation is still NP-complete, the same applies for the recognition of grid intersection graphs when given an L-representation (the same also trivially applies for $\{\mathsf{L}, \mathsf{J}\}$-representations).

Theorem 7. *The decision problem* Grid intersection graph recognition *is NP-complete even when given an* L*-graph representation.*

Proof. Once again, we will use Kratochvíl's construction, which uses clause gadgets, of which only satisfied clause gadgets are representable when restricted to grid intersection graphs. It is sufficient to show that it is possible to represent an unsatisfied clause gadget with L-shapes. This is shown in Fig. 2.10.

Therefore, G'_{φ} is an L-graph, and its representation can be constructed in polynomial time, which implies the theorem. □

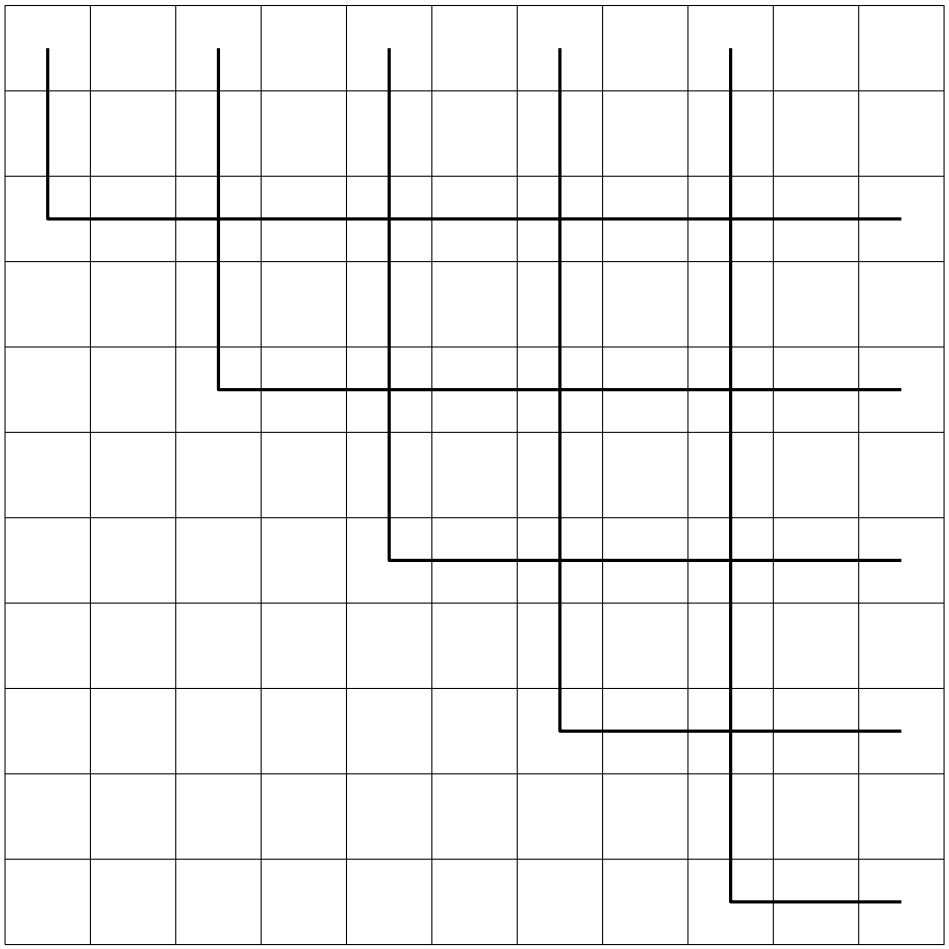

Figure 2.6: The graph K_5 with the grid overlay

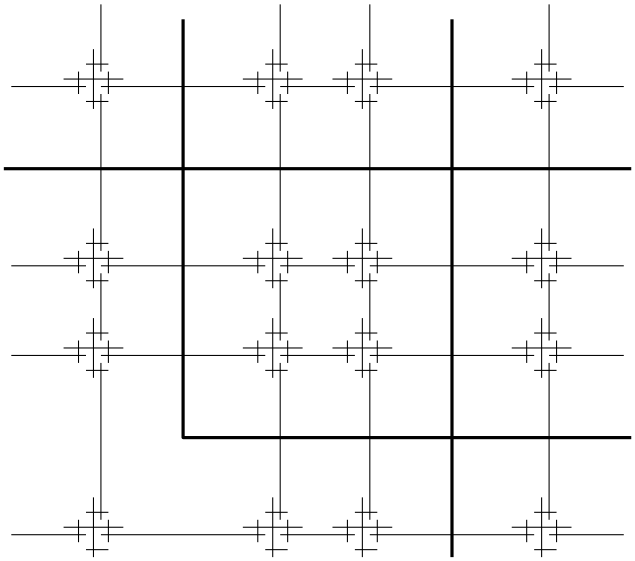

Figure 2.7: A part of the construction joining the grid with the K_5

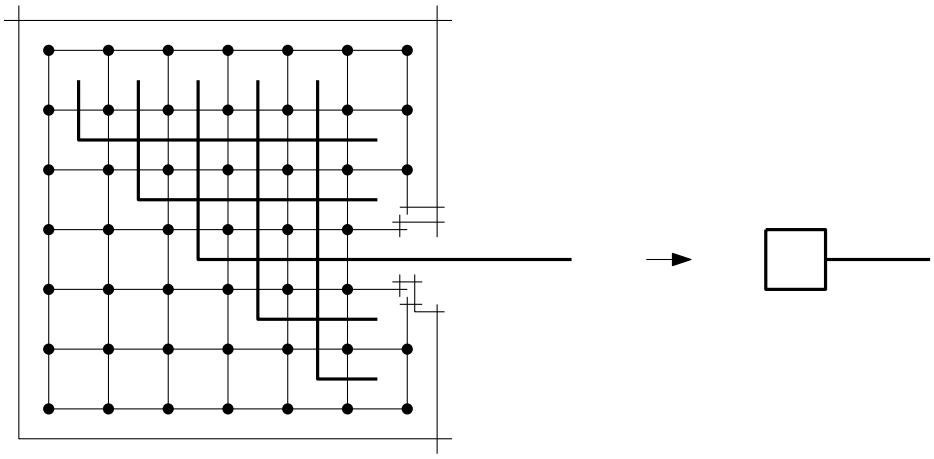

Figure 2.8: The pin construction

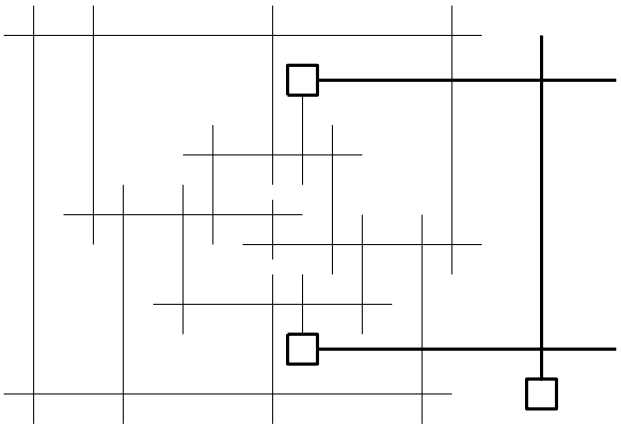

Figure 2.9: The clothespin construction

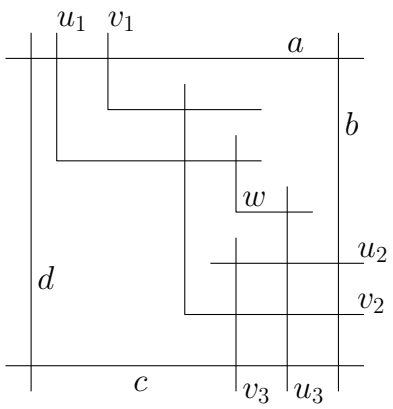

Figure 2.10: The unsatisfied clause gadget

2.3 Recognizing L-graphs algorithmically

In this section, we discuss an algorithm which creates an L-representation for every L-graph and rejects the input otherwise with time complexity $\mathcal{O}((n!)^2 \cdot n^2)$.

The following observation motivates the algorithm: in any L-representation, if two L-shapes a, b intersect, then it cannot happen that the bend of a is both lower than and more to the left than the bend of b. Hence for an L-graph $G = (V, E)$ with an L-representation R, we take a linear order $(V, <)$ of the bends from left to right. That is, for $u, v \in V : u < v$ if and only if the bend of u is more to the left than the bend of v. The linear order $<$ then induces a strict poset $(V, \prec)$, where $\prec$ is the transitive closure of the order $(u < v) \wedge \{u, v\} \in E$. The poset's order $u \prec v$ then represents the relation "the bend of vertex u must be higher than and more to the left than the bend of vertex v". Now, we observe that the order induced by the order of bends from top to bottom is another linear extension of $\prec$.

One possible way of generating these posets is taking all possible linear orders $<$ and then creating the poset $\prec$. Any linear extension of the generated strict poset is a possible candidate for a linear order on the y-axis. We are able to generate these linear extensions in time linear with respect to the number of linear extensions, using an algorithm by Oko and Nakano [29].

Now, for any order and its extension, it is easy to check in time $\mathcal{O}(n^2)$ whether the two orders can be extended into the L-representation. One possible algorithm is extending both the line segments of every vertex v so that the lines reach all of v's neighbors. Then, we check if the graph that is represented is the same as the original graph.

Proposition 8. *There exists an algorithm for recognizing* L*-graphs and generating the corresponding* L*-representation with time complexity* $\mathcal{O}((n!)^2 n^2)$.

Proof. We discuss the following algorithm:

Algorithm 1 (Recognition of L-graphs)**.** Input: a graph $G = (V, E)$.
For every linear order $<$ of V:

1. generate the strict poset $\prec$
2. for every linear extension $\lhd$ of $\prec$, check if there exists an L-representation with the orders of bend points on x and y axis as follows:
   (a) for every vertex v, extend the horizontal line segment as little as possible so that all neighbors of v that have their bend points more to the right are below the line segment
   (b) for every vertex v, extend the vertical line segment as little as possible so that all neighbors of v that have their bend points above v are to the left of the line segment
   (c) check if the graph induced by the representation is G and if so, return the representation

If no representation was found, G has no L-representation.

First, we show the correctness of the algorithm. The algorithm cannot return a representation of a different graph, hence if the algorithm with input G returns an L-representation, G is an L-graph. On the converse, if G is an L-graph, it has an L-representation R. Such representation can be generated, as the order $<$ based on R was surely used to generate $\prec$ in the algorithm, and $\lhd$ based on R is one of its linear extensions. The representation R also has to have its L-shapes extended at least as far as in the algorithm so that the required shapes intersect. Therefore the algorithm must find an L-representation.

Next, we analyze the time complexity of the algorithm. The outer for loop has to exhaust every linear order on V. As $|V| = n$, there are $n!$ distinct linear orders. During the inner loop, we first generate the poset, which can be done in time $\mathcal{O}(n^4)$. Then, we try every linear extension of the poset, there are $n!$ linear extensions in the worst case. Finally, with a given extension we can extend the line segments and check the graph in time $\mathcal{O}(n^2)$. This yields the time complexity $\mathcal{O}((n!)^2 n^2)$. □

3. Other problems on the classes

In this chapter we investigate other decision problems restricted to L-graphs or grounded L-graphs. More precisely, we show a polynomial-time algorithm for Clique both for L-graphs and grounded L-graphs. At the same time, we show NP-completeness of the problems Independent Set and 3-Coloring on L-graphs. Moreover, we discuss possible approximation algorithms for Coloring on grounded L-graphs.

3.1 Clique

As we mentioned in the first chapter, it is already known that Clique is in P for $\{\mathsf{L}, \mathsf{\lrcorner}\}$-graphs [21]. However, the restriction of the set of shapes makes it possible to reduce the time complexity.

The provided algorithm uses a previously known algorithm for Clique on permutation graphs.

Definition 13 (Permutation graph)**.** A permutation graph is an intersection graph of line segments whose endpoints lie at two parallel lines.

Equivalently, a permutation graph G is a graph on n vertices induced by a permutation $\pi\colon [n] \to [n]$ as follows: $G_\pi = ([n], \{\{i,j\} : i,j \in [n] \wedge i < j \wedge \pi(i) > \pi(j)\})$.

Theorem 9 (Clique on permutation graphs [30])**.** *There exists an algorithm solving* Clique *on permutation with time complexity* $\mathcal{O}(n \log n)$*.*

Proof. Given the permutation representation of G, we work with $\pi(1), \ldots, \pi(n)$ as a finite sequence. The maximum clique of G then corresponds to the longest decreasing subsequence of π.

Such longest decreasing subsequence can be found in time $\mathcal{O}(n \log n)$ [31]. □

Theorem 10 (Clique on grounded L-graphs)**.** *There exists an algorithm solving* Clique *when restricted to grounded* L*-graphs with time complexity* $\mathcal{O}(n^2 \log n)$ *when given a geometric representation.*

Proof. For every clique, there exists a rightmost vertex, and all horizontal line segments of the vertices of the clique must extend farther to the right than the vertical line segment of the right-most vertex.

This motivates the following algorithm: For every vertex v, we take the graph induced by the vertices whose horizontal lines intersect the vertical line segment of v – this corresponds to the choice of v as the rightmost vertex. This graph is a permutation graph on k vertices, for which we are able to find a maximum clique in time $\mathcal{O}(k \log k)$ by Theorem 9. Then, we take the maximum of these maximum cliques to be the result.

Given that there are n possible choices for the right-most vertex and in the induced graph, there may be $n - 1$ vertices, the worst-case time complexity is $\mathcal{O}(n^2 \log n)$ □

Theorem 11 (Clique on L-graphs)**.** *There exists an algorithm solving* Clique *when restricted to* L*-graphs with time complexity* $\mathcal{O}(n^3 \log n)$ *when given a geometric representation.*

Proof. We will proceed similarly as with grounded L-graphs. For every clique, there exists a topmost vertex, and the vertical segments of the clique's vertices must extend higher than the horizontal line segment of the top-most vertex.

Hence, for every vertex v, we create the graph induced by the vertices whose vertical line segments intersect the horizontal line segment of v. This graph is a grounded L-graph. For such graphs, there exists an algorithm for Clique running in time $\mathcal{O}(n^2 \log n)$ by Theorem 10. We take the maximum of the maximum cliques as the result.

Similarly as in the previous theorem, we have n choices for the top-most vertex, which leaves us with a graph on at most $n-1$ vertices, so our total time complexity is $\mathcal{O}(n^3 \log n)$. □

3.2 Independent Set

The situation with Independent Set is similar. We already know that restriction to B_0-VPG (or 2-DIR) graphs still results in a problem which is NP-complete.

We present an alternative proof on NP-completeness based on a reduction from the decision problem Planar Monotone 3-SAT.

Definition 14 (Monotone rectilinear representation)**.** Let φ be a CNF formula with a set of clauses C over a set of variables X such that

1. every clause contains at most three variables,
2. each clause contains only positive literals or only negative literals,
3. the bipartite graph $G_\varphi = (X \cup C, \{\{x, c\} : x \in c \in C \vee (\neg x) \in c \in C\})$ is planar.

The formula's *monotone rectilinear representation* is a planar drawing of the graph G_φ such that

- all variables and clauses are drawn as rectangles,
- the edges connecting the variables to their clauses are vertical segments,
- the drawing is crossing-free,
- all positive clauses are drawn above the variables,
- all negative clauses are drawn below the variables,
- there exists a horizontal line which intersects all rectangles corresponding to the variables.

Definition 15 (Planar Monotone Rectilinear 3-SAT)**.**

Planar Monotone Rectilinear 3-SAT
Instance: A monotone rectilinear representation of a CNF formula φ. **Question:** Is φ satisfiable?

Theorem 12 (de Berg & Khosravi, 2010 [24]). *The decision problem* Planar Monotone Rectilinear 3-SAT *is NP-complete.*

Theorem 13 (Independent Set on L-graphs). *The decision problem* Independent Set *is NP-complete when restricted to* L*-graphs even when given a geometric representation.*

Proof. It is easy to see the problem is in NP – the certificate is the set of vertices in the independent set.

To show that the problem is NP-hard, we will reduce from Planar Monotone Rectilinear 3-SAT. We are given the monotone rectilinear representation of the formula. Without loss of generality, we may assume that for each variable, all the line segments connecting its rectangle to the rectangles representing its positive occurrences in clauses lie on the same vertical line, and the same applies for the negative occurrences.

Moreover, the line for negative instances is to the right from the line of the positive instances and no two rectangles representing a positive clause containing the same variable have the same y-coordinate. Otherwise, we perturb them.

The reduction is the same as in the case of general Independent Set. For a formula $\varphi = \bigwedge_{i=1}^{m} C_i$, we create graph $G_\varphi = (V, E)$. The graph G_φ has a vertex for each positive or negative occurrence of a variable in a clause. For every clause $C_i = \bigvee_{j=1}^{k} l_{i,j}$, we connect all of its occurrences of variables with edges. Then, for every two literals which represent the same variable, with one being the negation of the other, we add an edge between them. This ensures we may never choose both x_i and $\neg x_i$ as the variables of the independent set.

This can also be represented as an L-graph, as shown in Figures 3.1, 3.2 and 3.3.

Finally, we show that the formula φ is satisfiable if and only if the graph G_φ has an independent set of size m.

For the forward implication, let φ be a satisfiable formula with a satisfying assignment a. As the formula is satisfied by a, for every clause there exists at least one true literal. We may choose these literals to represent the independent set (if there are more true literals in a single clause, we choose one arbitrarily), as it cannot happen that we would choose two literals connected by an edge. There are two types of edges: edges between literals in the same clause, which are handled by choosing exactly one literal per clause, and edges between positive and negative instances of the same variable, of which we can only choose one, as precisely one of the instances has true value.

For the converse, let there exist an independent set $I \subset V$ of size m. This means that there exists a single variable which was chosen in every clause. Take all variables v_i such that there exists a literal $l_{j,k} \in I$ such that $l_{j,k} = v_i$. We will set these variables to 'true' and the rest (including v_i such that $l_{j,k} \in I$ and $l_{j,k} = \neg v_i$) to 'false'. It may not happen that this would produce an unsatisfying assignment, as it cannot happen that there would be two literals, one the negation of the other, as they would not form an independent set. Therefore, we have a satisfying assignment and the formula is satisfiable. □

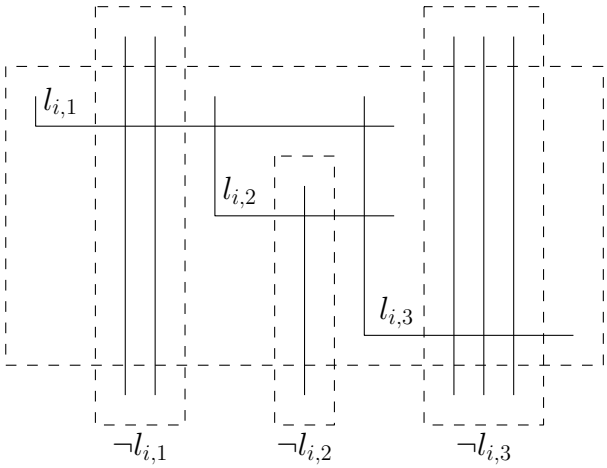

Figure 3.1: The positive clause gadget for Independent Set

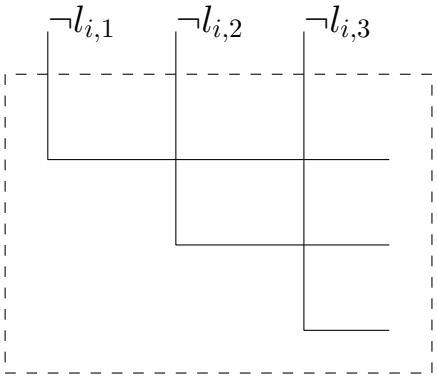

Figure 3.2: The negative clause gadget for Independent Set

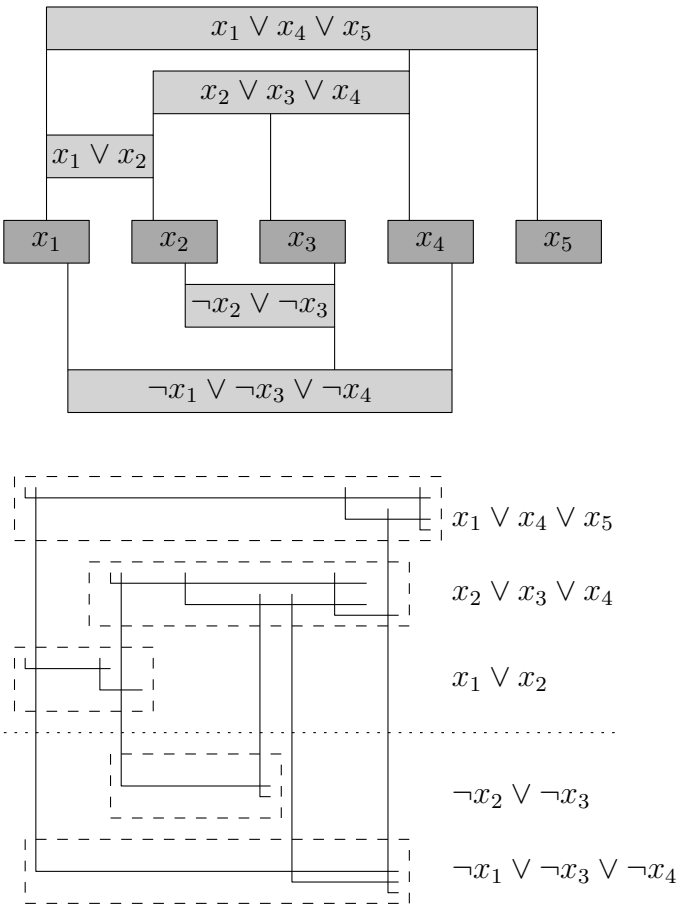

Figure 3.3: An example of reduction from Planar Monotone Rectilinear 3-SAT to Independent Set

3.3 Coloring

3.3.1 NP-completeness on L-graphs

Theorem 14 (3-Coloring L-graphs)**.** *The decision problem 3-*Coloring *is NP-complete when restricted to* L*-graphs even when given a geometric representation.*

Proof. We will show that Planar Monotone Rectilinear 3-SAT reduces to 3-Coloring on L-graphs. Given an instance of Planar Monotone Rectilinear 3-SAT, we will construct an instance of 3-Coloring as follows.

As with the reduction to Independent Set, we may assume that all line segments from the rectangles representing positive instances of variables lie on a single line for each variable, the same applies for the negative instances and the line for negative instances is to the right from the line of the positive instances.

For every variable x_i, we create two vertices $x_i, \overline{x_i}$ joined by an edge, all connected to another vertex denoted by B. The vertex B also forms a triangle with vertices T, F. The colors of these two vertices then correspond to the truth values true and false respectively. We also add another vertex T' which forms a triangle with the vertices B, F.

Every positive clause (i.e. above the line) corresponds to a *positive clause gadget*, while every negative clause corresponds to a *negative clause gadget*. These gadgets are shown in Figures 3.4, 3.5.

Next, we need to connect the clause gadgets. Every heightwise maximal positive clause gadget's two vertices denoted by $\neg T$ in the figure are both connected by an edge to the vertex T. At the same time, we add the two edges between the vertex T' and the two vertices denoted by $\neg T$ in every heightwise minimal negative clause gadget. If there are $k \geq 2$ clause gadgets between the vertices a, b or b, c in the representation, then we create another vertex $\hat{T}$ that forms a triangle with one of $\overline{\neg T}, \widehat{\neg T}$ and add the vertices $\neg T$ of each clause as the neighbors of $\hat{T}$. For all other clause gadgets, we identify its vertices of $\neg T$ with one of $\overline{\neg T}, \widehat{\neg T}$ based on the position of the clause gadget with respect to the clause gadget above it.

An example of the reduction can be seen in Figure 3.7.

Next, we show that a 3-coloring of a clause gadget exists if and only if the clause is satisfied (i.e. at least one of its variables has a color which is the same as the color of the vertex T).

From now on, we will use T, F and B to denote the colors of vertices T, F and B.

The gadgets use two OR gadgets (Fig. 3.6) to join the colors. If both s, t are colored F, then the vertices q, r must be colored using colors T and B, hence vertex p must colored F. At the same time, if at least one of s, t is not colored F, then we can use colors F and B and the vertex p can be colored T.

First, we assume that the 3-coloring exists. In every such coloring, the color of the vertex $a \vee b \vee c$ must be T. This implies that at least one of the vertices c or $a \vee c$ are not colored F, and therefore at least one of the vertices a, b, c is not colored F. Because all vertices representing variables can only be colored T or F, it simply follows that the clause is satisfied.

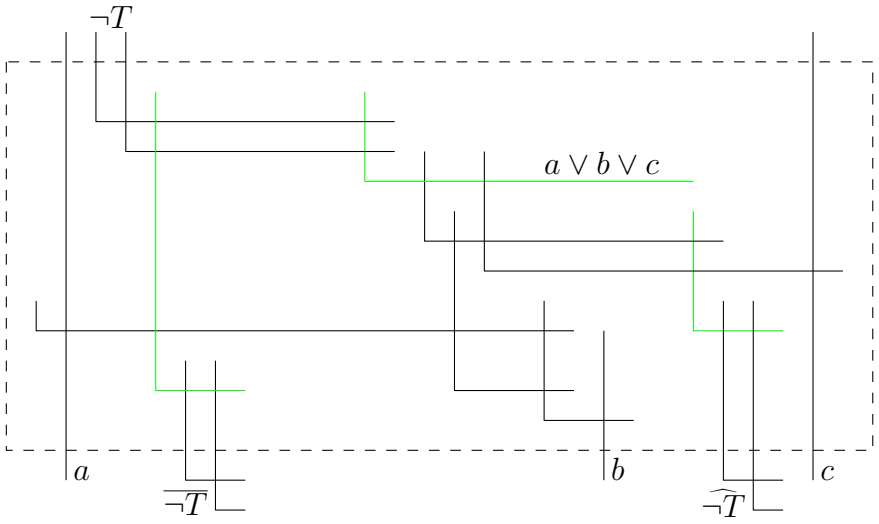

Figure 3.4: The positive clause gadget for 3-Coloring

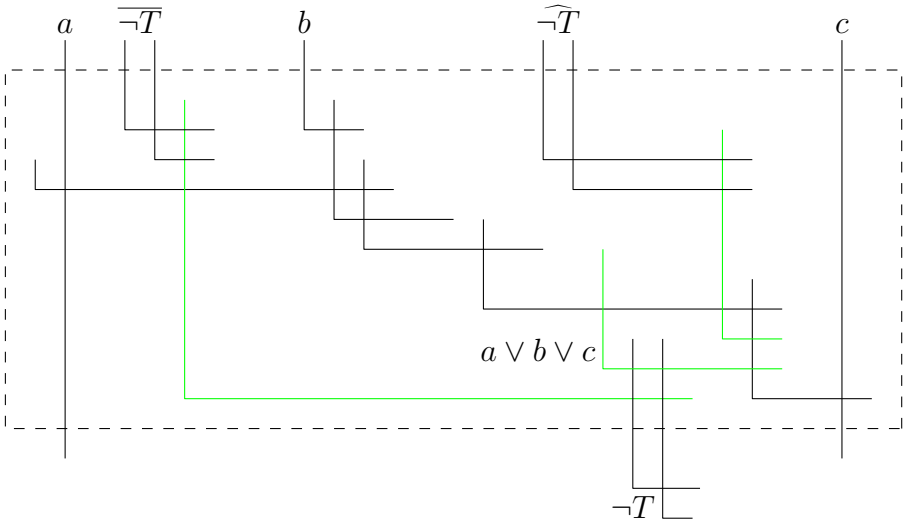

Figure 3.5: The negative clause gadget for 3-Coloring

For the converse, if the clause is satisfied, then we can color the variables by their truth values in the satisfying assignment: a true variable will be colored T, while a false variable will be colored F. As the clause is satisfied, then at least one variable is colored by T, and by the argument above, the vertex $a \vee b \vee c$ can be colored T.

As the argument does not change for the construction as a whole, we conclude that 3-Coloring L-graphs is NP-complete. □

Theorem 15 (3-Coloring triangle-free L-graphs)**.** *The decision problem 3-*Coloring *is NP-complete when restricted to triangle-free* L*-graphs even when given a geometric representation.*

Proof. We use the same reduction as in the case of general 3-Coloring and transform every vertex into a triangle-free circle graph with two vertices m, n which have the same color in every 3-coloring of the graph. Then all vertices which intersect the original L-shape's horizontal line segment are now the neighbors

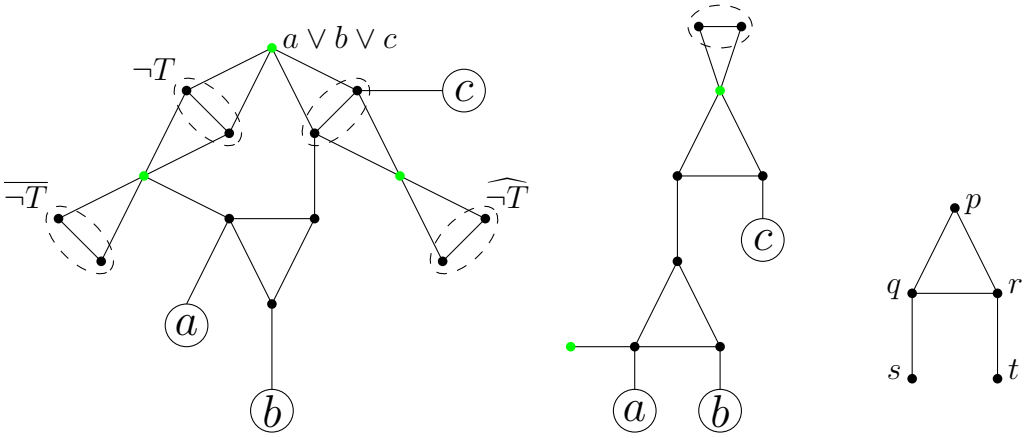

Figure 3.6: The clause gadget, the union of two OR gadgets, and a single OR gadget

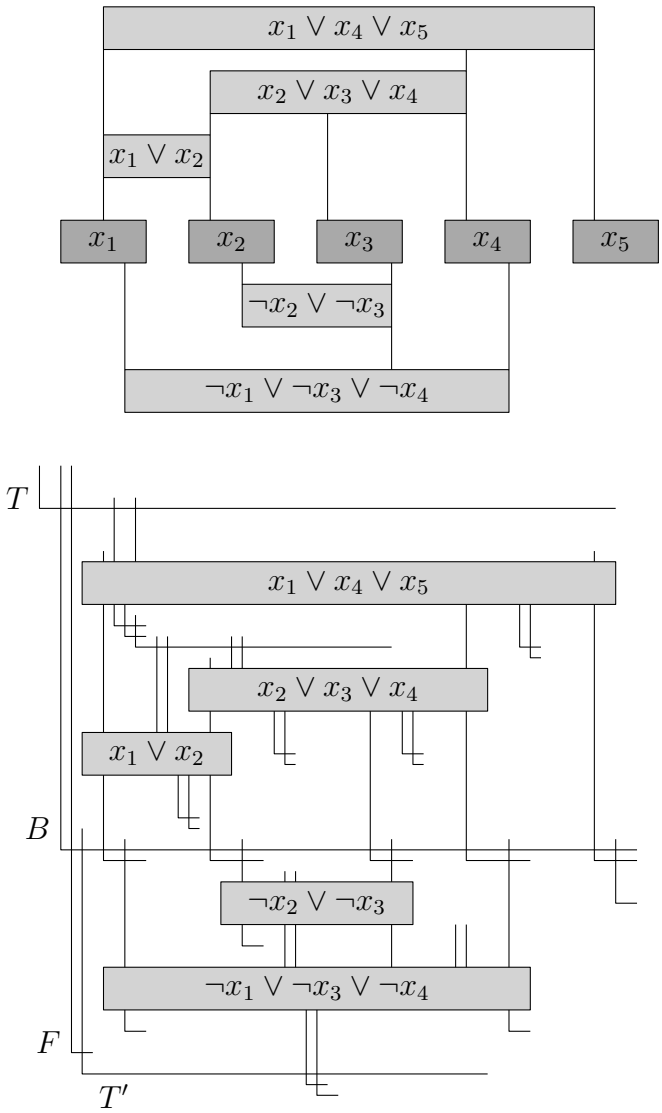

Figure 3.7: An example of the 3-Coloring reduction

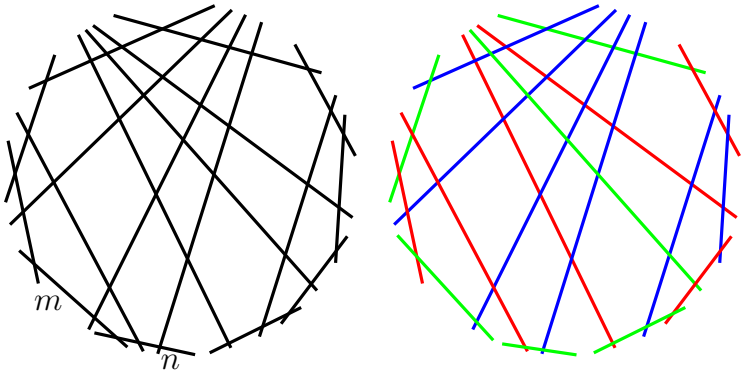

Figure 3.8: A 4-colorable circle graph and its 3-colorable circle subgraph created by removing the edge $\{m, n\}$

of the vertex n and all vertices intersecting the original L-shape's vertical line segment are the neighbors of the vertex m. As the subgraph is a circle graph, it has a representation using only L-shapes with both of their endpoints on a circle. Therefore we can arbitrarily extend the horizontal line segment of n and the vertical line segment of m.

We construct the subgraph in two steps. First, we take a triangle-free circle graph H which needs four colors. In our case, H is the left part of Figure 3.8. Then, we remove the edge $\{m, n\}$ to create the graph H' so that it is still a circle graph and the graph is now 3-colorable, as in the right part of Figure 3.8. We observe that vertices m and n must have the same color in every coloring of the graph H' – if there existed a 3-coloring c of H' so that $c(m) \neq c(n)$, c would also be a 3-coloring of the graph H, which is a contradiction, as we know that H is not 3-colorable.

The reduction then works the same and is still polynomial, as every vertex is now extended into 17 vertices. □

3.3.2 Approximation on grounded L-graphs

As Coloring is NP-complete on circle graphs [25], we also obtain NP-completeness on grounded L-graphs. Therefore, it is natural to ask if there exist any approximation algorithms with a constant approximation ratio. The linearity of the natural order of the vertices in representation seems to indicate that First Fit could be a reasonable strategy. It is easy to see that the First Fit algorithm finds a coloring with at most $\Delta + 1$ colors, where Δ is the maximum degree of a vertex in the colored graph.

However, it turns out that we cannot do any better – there exist graphs which attain the bound.

Lemma 16. *For every $k \geq 2$ there exists a bipartite graph G_k with $n = 2^{k-1}$ vertices, a maximum degree $\Delta \in \Omega(\log n)$ and a grounded L-representation such that the First Fit approach uses $k = \Delta$ colors.*

Proof. We construct the graph inductively: it is helpful to also create G_1, which simply contains a single vertex (and can be colored using a single color). As for

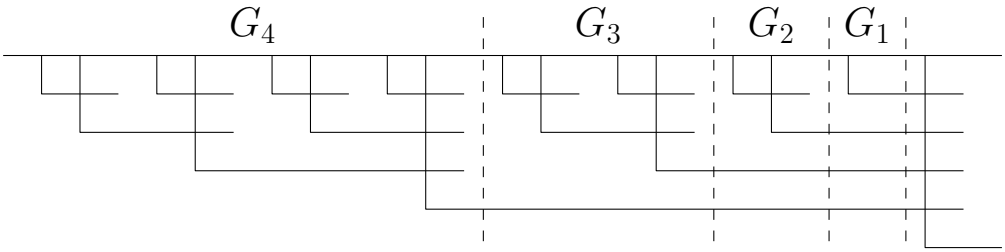

Figure 3.9: The construction of G_5 in Lemma 16

the base case $k = 2$, we construct such a graph using two vertices connected to each other.

Now we proceed with the induction step – assume we already constructed G_i for every $i \in [k-1]$. Then we construct G_k by drawing G_l for $l = k-1$ to $l = 1$ left-to-right so that G_l is below G_{l-1} and the vertex of G_l colored by l can be extended arbitrarily far to the right without intersecting any other vertex from $G_p : p < l$. Then we extend the "color-maximizing" vertices and intersect them all with a new vertex, which will be colored by color k.

In total, we use $1 + \sum_{l=1}^{k-1} 2^{l-1} = 1 + 2^{k-1} - 1 = 2^{k-1}$ vertices. □

It is not difficult to see that sometimes, we do not reuse colors even though we could – namely, if we are currently trying to color a vertex v, which joins two or more previously disconnected components. As an example, let us have two components where one has two neighbors of v with colors 1 and 3 and the other has also two neighbors with colors 2 and 4. In this case, it would make sense to greedily permute the colors in one of the classes so that the neighbors of v only use 2 colors.

The main tool we use is the construction of connected components so far. For every vertex, we look at the connected components it intersects (there exists at most one component which is not connected to the vertex completely – the lowest one). Now, for every component, we look at the colors which have been used already by the intersected vertices and permute them so that the k used colors are now numbered $1, \ldots, k$. Then, we use the lowest available color number.

Algorithm 2 (Permuted First Fit)**.** Input: a grounded L-graph with vertices $v_1, \ldots, v_n$ ordered left-to-right.

Output: coloring of the vertices.

1. For $i = 1$ to n:
   (a) If v_i does not intersect any other vertex, create a new component and color v_i with color 1
   (b) If v_i intersects precisely one component, choose v_i's color according to the First Fit scheme (take the lowest color which is not used by v_i's neighbors)
   (c) Otherwise, for every component v_i intersects, permute the colors of the component so that the k colors of vertices which have a nonempty intersection with v_i are numbered $1, \ldots, k$, then color v_i with color $k+1$ and create a new component by joining the components and adding v_i

Let us remark that while First Fit works as an online algorithm, Permuted First Fit does not. In online coloring algorithms, the color of any given vertex cannot change, however such change may happen in the Permuted First Fist.

On the other hand, this would pose no issue if we lowered our expectations on the online algorithm. If it was not necessary to report the color of the vertex, only the vertices with the same color, this would still be an online algorithm, but for a different problem.

However, it turns out that the modification does not help much.

Lemma 17. *For every $k \geq 2$ there exists a tripartite graph G'_k with $n = 1 + 2^{k-1}$ vertices, a maximum degree $\Delta \in \Omega(\log n)$ and a grounded L-representation such that the Permuted First Fit approach uses $k = \Delta$ colors.*

Proof. We will slightly modify the graph G_k from Lemma 16 by adding a new vertex v', which lies above all other vertices and intersects them all.

It is easy to see that the graph is tripartite, as it was bipartite and we have added a single vertex, which cannot increase the chromatic number by more than one.

At the same time, the maximum degree obviously increases by one. By adding the vertex, we have successfully removed any possibility of permuting the vertices, as all vertices are in the same component as v'. This means we use the same number of colors as if we used First Fit on $G'_k - v'$ and v' has to use another color, from which we obtain that we use $\Delta - 1 + 1$ colors. □

Conclusion

In the thesis we studied various decision problems restricted to three classes of intersection graphs: L-graphs, grounded L-graphs, and $\{\mathsf{L}, \lrcorner\}$-graphs. The first problem we discussed was Recognition, precisely L-graph recognition and $\{\mathsf{L}, \lrcorner\}$-graph recognition. In both cases we proved that the problem is NP-complete answering an open question of Felsner et al.[11]. Next, we focused on Clique on L-graphs and grounded L-graphs and provided polynomial algorithms for both classes with better time complexity than the algorithms by Middendorf and Pfeiffer [21]. In the case of Independent Set, we showed that it is NP-complete on L-graphs. Finally, we showed that 3-Coloring is NP-complete even on triangle-free L-graphs and discussed possible approximation algorithms for Coloring of grounded L-graphs.

There are still many open questions surrounding these problems. We will focus on three, which are closely related. First, is it possible to use the approach used in proving NP-completeness of L-graph recognition and $\{\mathsf{L}, \lrcorner\}$-graph recognition to show NP-completeness of other subclasses of B_k-VPG graphs? For example, the case of $\{\sqcup\}$-graphs seems to have an issue, where we need a pin-like construction, which works in two separate "modes": either it enables the main vertex to arbitrarily extend its horizontal segment and both of its vertical segments are bounded in the "pin heads" or it enables the main vertex to arbitrarily extend precisely one of the two vertical segments and has both of its bends inside a single "pin head".

Second, what is the complexity of Grounded L-graph recognition? Both a polynomial algorithm and the NP-completeness of the problem would be interesting, as the polynomial algorithm could possibly be extended to some other forbidden vertex orders and the same could apply for the polynomial reduction.

Third, what is the complexity of 3-Coloring outerstring graphs or grounded L-graphs? Both L-graphs and string graphs still retain NP-completeness of 3-Coloring and either result might shine a light on strength of the additional restriction of all shapes touching a line with one of their endpoints.

Bibliography

[1] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974.

[2] Kellogg S. Booth and George 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, 1976.

[3] Csaba P. Gabor, Kenneth J. Supowit, and Wen-Lian Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3):435—473, 1989.

[4] Donald J. Rose, Robert E. Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.

[5] Fănică Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5):181–188, 2000.

[6] Martin Pergel. Recognition of polygon-circle graphs and graphs of interval filaments is NP-complete. In *Graph-Theoretic Concepts in Computer Science*, pages 238–247. Springer, 2007.

[7] Jan Kratochvíl. String graphs. II. Recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 1991.

[8] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003. Special Issue on STOC 2002.

[9] Jan Kratochvíl and Jiří Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994.

[10] Steven Chaplick, Vít Jelínek, Jan Kratochvíl, and Tomáš Vyskočil. Bend-bounded path intersection graphs: Sausages, noodles, and waffles on a grill. In *Graph-Theoretic Concepts in Computer Science*, pages 274–285. Springer, 2012.

[11] Stefan Felsner, Kolja Knauer, George B. Mertzios, and Torsten Ueckerdt. Intersection graphs of L-shapes and segments in the plane. *Discrete Applied Mathematics*, 206:48–55, 2016.

[12] Andrei Asinowski, Elad Cohen, Martin Charles Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. Vertex intersection graphs of paths on a grid. *Journal of Graph Algorithms and Applications*, 16(2):129–150, 2012.

[13] Steven Chaplick and Torsten Ueckerdt. Planar graphs as VPG-graphs. In *Graph Drawing*, pages 174–186. Springer, 2013.

[14] Therese Biedl and Martin Derka. 1-String B_2-VPG Representation of Planar Graphs. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 141–155. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.

[15] Daniel Gonçalves, Lucas Isenmann, and Claire 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.

[16] Vít Jelínek and Martin Töpfer. On grounded L-graphs and their relatives. *The Electronic Journal of Combinatorics*, 26(P3.17), 2019.

[17] Pavol Hell, Bojan Mohar, and Arash Rafiey. Ordering without forbidden patterns. In *Algorithms - ESA 2014*, pages 554–565. Springer, 2014.

[18] Daniel Heldt, Kolja Knauer, and Torsten Ueckerdt. Edge-intersection graphs of grid paths: The bend-number. *Discrete Applied Mathematics*, 167:144–162, 2014.

[19] Kathie Cameron, Steven Chaplick, and Chính T. Hoàng. Edge intersection graphs of L-shaped paths in grids. *Discrete Applied Mathematics*, 210:185–194, 2016.

[20] Martin Pergel and Paweł Rzążewski. On edge intersection graphs of paths with 2 bends. In *Graph-Theoretic Concepts in Computer Science*, pages 207–219. Springer, 2016.

[21] Matthias Middendorf and Frank Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Mathematics*, 108(1):365–372, 1992.

[22] Jan Kratochvíl and Jaroslav Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 031(1):85–93, 1990.

[23] Prosenjit Bose, Paz Carmi, Mark J. Keil, Anil Maheshwari, Saeed Mehrabi, Debajyoti Mondal, and Michiel Smid. Computing maximum independent set on outerstring graphs and their relatives. In *Algorithms and Data Structures*, pages 211–224. Springer, 2019.

[24] Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In *Computing and Combinatorics*, pages 216–225. Springer, 2010.

[25] Michael R. Garey, David S. Johnson, Gary L. Miller, and Christos H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.

[26] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[27] Herbert Grötzsch. Zur Theorie der diskreten Gebilde, VII: Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 8:109–120, 1959.

[28] Jan Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

[29] Akimitsu Ono and Shin-ichi Nakano. Constant time generation of linear extensions. In *Fundamentals of Computation Theory*, pages 445–453. Springer, 2005.

[30] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*, chapter 7 - Permutation graphs, pages 157–170. Elsevier, 2004.

[31] Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975.

List of Figures