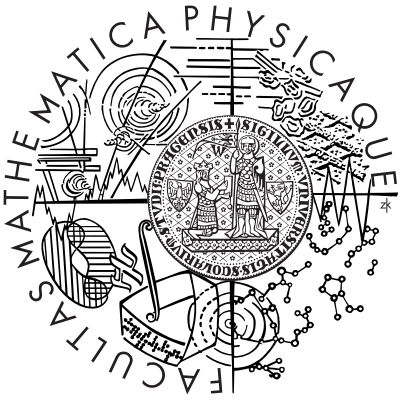
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Tomáš Vyskočil

Graph Drawing: Visualization and Geometric Representations of Graphs and Networks

Department of Applied Mathematics

Supervisor of doctoral thesis: *Prof. RNDr. Jan Kratochvíl, CSc.*

Study programme: *I4 - Discrete models and algorithms*

Prague 2015

Acknowledgement

My special thanks belong to my advisor, Prof. Jan Kratochvíl, for leading me through my studies and research.

I am grateful to colleagues from my department Pavel Klavík, Vít Jelínek for fruitful discussions concerning results presented in this thesis and also to guests of our department Pavol Hell, Mathew Francis and Steven Chaplick.

Last but not least, I would like to thank all the members and the students of the Department of Applied Mathematics for the wonderful environment, support, and excellent background without which I could hardly write this thesis.

I declare that I carried out this doctoral 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 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature

Název práce: Kreslení grafů: Vizualizace a geometrické reprezentace grafů a sítí

Autor: Tomáš Vyskočil

Katedra: Katedra applikované matematiky

Vedoucí: Prof. RNDr. Jan Kratochvíl, CSc., KAM

Abstrakt: Tato práce se věnuje studiu reprezentací grafů. V kapitolách 2–4 studujeme průnikové grafy v rovině a v kapitole 5 studujeme problémy modifikací grafů pomocí jednoduchých operací.

V části věnované průnikovým grafům se věnujeme následujícím. Ukážeme, že částečné 2-stromy jsou průnikové grafy úsečkových grafů. Dále ukážeme složitost rozpoznání průnikových grafů k lomených cest na mřížce a průnikových grafů ostrovů v rozšířené mřížce.

V části věnované modifikacím grafů ukážeme FPT-algoritmus který řeší problém, zda můžeme získat rovinný graf ze vstupního grafu pomocí nejvýše k kontrahovaných hran a zobecnění tohoto problému.

Klíčová slova: teorie grafů, reprezentace grafů, kombinatorika

Title: Graph Drawing: Visualization and Geometric Representations of Graphs and Networks

Author: Tomáš Vyskočil

Department: Department of applied mathematics

Supervisor: Prof. RNDr. Jan Kratochvíl, CSc., KAM

Abstract: This thesis is devoted to studying of representations of graphs. In Chapters 2–4 we study intersection graphs in the plane. In Chapter 5 we consider problems of modifications of graphs.

Regarding intersection graphs, we prove that all complements of partial 2-trees are intersection graphs of segments. We show complexity of recognition of intersection graphs of path in the grid with bounded number of bends and intersection graphs of connected regions (we call them islands) in the extended grids.

In the part devoted to modification problems we present a fixed-parameter tractable (FPT) algorithm which answers the question whether a given graph can be made planar with at most k contractions and we also provide generalization of this problem.

Keywords: graph theory, graph representations, combinatorics

List of publications relevant to thesis

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

[2] James Abello, Pavel Klavík, Jan Kratochvíl and Tomáš Vyskočil, MSOL restricted contractibility to planar graphs, Parameterized and Exact Computation (2012), 194–205.

[3] Mathew C. Francis, Jan Kratochvíl and Tomáš Vyskočil, Segment representation of a subclass of co-planar graphs, Discrete Mathematics Vol. 312, No. 10 (2012), 1815–1818.

[4] Michael D. Coury, Pavol Hell, Jan Kratochvíl and Tomáš Vyskočil, Faithful representations of graphs by islands in the extended grid, LATIN 2010: Theoretical Informatics (2010), 131–142.

Contents

Introduction

In this thesis we study different representations of graphs. For practical applications it is crucial to visualize graphs on paper or computer screen. Therefore, it is not surprising that this area is so broad that no single book can contain it all. In this thesis we consider several topics which are of special interest to the author.

Intersection Graphs. A large portion of the thesis is about intersection graphs which turns out to be very important in several disciplines including biology, VLSI design and many more. Intersection graph theory studies classes of graphs where vertices are represented by sets and sets are adjacent if they intersect. Several important problems, which are hard for general graphs turn out to be efficiently solvable for some intersection classes of graphs which makes it very practical. As an example we can mention interval graphs, which is a very useful class of graphs (for example to model DNA maps [68]). There are several problems, hard for many classes of graphs, but which turn out to be simple for interval graphs, for example, the maximal clique problem or the problem of recognition. There are several other interesting intersection graph classes and we introduce some of them. Unfortunately, most of our results are rather negative, meaning that corresponding problems are not polynomially solvable.

Modifications of Graphs. Apart from intersection graphs, we study modifications of graphs. By modification of graphs we mean problems of transforming graphs by using deletion, addition or contraction of edges or vertices. A typical question can be "What is the minimal number of edges to delete so that the graph becomes planar?" It turns out that most of graph problems can be described by the language of modifications of graphs. One possible example is the graph connectivity problem which can be also seen as finding the minimal number of vertices to be removed so that the graph becomes disconnected. The maximum clique problem which can be seen as problem

of finding the minimal number of vertices so that the graph becomes a complete graph is yet another example. In this work we explore modifications of graphs by contractions. We show algorithmic results, specifically a fixed parameter algorithm and a hardness reduction which adjusts the importance of the fixed parameter algorithm.

Structure of the Thesis. The first three chapters are devoted to intersection graphs and the last chapter introduces some problems of modifications of graphs. In Chapter 1 we introduce our notation and basic definitions from graph theory, computational complexity and intersection graph theory. In Section 2 we consider intersection graphs of segments where we describe related works and prove that the complements of partial 2-trees are contained in segment graphs. The other area of interest is so called VPG graphs which we study in Section 3, which are intersection graphs of path in the grid. We specifically study there B_k-VPG graphs which are graphs with at most k bends. We show related results and prove that the recognition of B_k-VPG graphs in B_{k+1}-VPG graphs is NP-complete and we also show that there is no inclusion between B_k-VPG (for any $k > 1$) and intersection graphs of segments. Chapter 4 is devoted to island representations of graphs. We show complexity of recognition of such graphs. In Chapter 5 we introduce modifications of graphs and present the fixed parameter algorithm for contractibility problem and its generalizations.

Chapter 1

Notation and Basics

1.1 Introduction to Graph Theory

One of the first scientists interested in graph theory was probably Leonhard Euler, and since then graph theory has grown into broad discipline with many branches. In this section we describe several basic terms from graph theory which will be of great use later.

An ordered pair $G = (V, E)$ is called *graph* if $E \subseteq \binom{V}{2}$, where V is a set of elements also commonly referred to as vertices and E is a set of elements commonly known as edges. By $V(G)$ we denote the set of vertices of graph G and similarly, $E(G)$ represents the set of edges. We often omit parameter G if the graph is clear from context. Very commonly n refers to the number of vertices and m to the number of edges, which are often used without mentioning. Notation used to denote an edge is $e = \{u, v\}$, where u and v are called the end-vertices of edge e. It is common to use uv to refer to edge $\{u, v\}$, and we use it here too. Vertices u and v of G are adjacent if there is edge uv in G. The set of neighbors of vertex v contains all vertices adjacent to v, and we denote it $N(v)$. Notation $N^+(v) := N(v) \cup \{v\}$ is also very common.

The graph $H = (U, F)$ is called *subgraph* of graph $G = (V, E)$ if $U \subseteq V$ and $F \subseteq E$. If for every tuple of vertices u,v in U there is edge uv in E which implies uv is in F, then we call H an *induced subgraph*. Let U be subset of vertices of G, then $G[U] = (U, \binom{U}{2} \cap E)$ is called the *graph induced* by vertex set U. You can notice $G[U]$ is an induced subgraph.

A sequence of vertices $v_1, e_1, v_2, e_2 \ldots, e_{k-1}v_k$ is called a *walk* if for every i

there is edge $e_i = v_i v_{i+1}$. If every vertex is present in the walk sequence just once we call it a *path*. A *cycle* is a walk where no vertex is repeated except for $v_1 = v_k$. The *length* of the path, walk and cycle is the number of edges in the sequence defining them. A graph without cycles is called *acyclic* graph. An acyclic connected graph is called a *tree*.

There are some well-known categories of graphs. Graph $G = (V, E)$ is called a *complete graph* if $E = \binom{V}{2}$. A complete graph with n vertices is denoted by K_n. A graph which is formed by path of length n is denoted by P_n and analogically a graph formed by cycle of length n is denoted by C_n. Graph G is called *bipartite* if the set of vertices $V(G)$ can be partitioned into two sets such that in $E(G)$ there is no edge connecting two vertices from the same set.

Graph G is *connected* if for every tuple of vertices u and v from $V(G)$ there is a path which starts in vertex u and ends in v. A *degree* of vertex is the number of edges which has v as an end-vertex.

The *complement* of graph $G = (V, E)$ is graph $\overline{G} = (V, \binom{V}{2} \setminus E)$.

The *oriented graph* is a graph where edges are not sets of vertices but tuples of vertices, for example, an edge from u to v is denoted as (u, v) and it is not the same edge as (v, u).

1.2 Basics of Computational Complexity

The center-point of computational complexity is algorithm. Although an algorithm is an elementary structure, its definition is not simple and typically is made using Turing Machines, even though it is not the only way. We present here just a brief introduction, and this introduction will not be self-contained. Its purpose is rather to remind a reader basic concepts. For more information on the topic we recommend to look into [5].

Algorithm. In this work we use Turing Machine model for purpose of NP-reductions. And for algorithms in Chapter 5 we use RAM (Random Access Memory) model. All algorithms we present here are decision algorithms which means that they always terminate and either return "accept" (or 1) or "reject" (or 0).

A (decision) algorithm defines a particular *language* (or decision problem) which is a set of *words* and words are a set of *letters* and letters are elements of an *alphabet* and the alphabet is a set of symbols (we can always assume

the alphabet of 0 and 1). Algorithm A solves language L if for every word w in L algorithm A "accepts" and otherwise "rejects".

The running time of algorithm is a function which returns a maximal number of steps of algorithm when it runs on inputs of size n. Typically its running time is not presented as a particular function, but it is rather given in so called *big O* notation, which is just an upper bound on running time and ignores a constant factor.

Complexity classes. There are basic classes of languages depending how hard it is to solve these languages. For purpose of this work the most essential classes are P and NP. Class P is a class of all languages which can be solved by algorithms whose running time is bounded by a polynomial function, such algorithms are called *polynomial algorithms*. Class NP contains language L if the following holds. Word w is in L if and only if there exists polynomial certificate c and polynomial algorithm A such that A "accepts" c and w.

Parametrized Complexity. Finally, we introduce parametrized complexity. This field was developed in order to solve fast problems which are intractable (NP-hard problems), which at first might sound like a contradicting claim. The idea is to introduce a new parameter and it is assumed to be a constant, which means we do not solve a general problem but a subset of the problem which depends on the parameter. Formally, the *parametrized problem* is language $L \subseteq \Sigma^* \times \mathbb{N}$, where the first component represents language, where Σ^* is a set of all words in alphabet Σ, and the second component represents a parameter. A parametrized problem is *fixed-parameter tractable* if word $(x, k) \in L$ can be decided in running time $f(k) \cdot |x|^{\mathcal{O}(1)}$, where f is an arbitrary function depending on k.

1.3 Basics of Intersection Graph Theory

In this chapter we define what intersection graphs are and introduce important intersection graph classes.

Concept. Let $\mathcal{M}$ be a collection of sets, where we assume that most of the sets are subsets of the Euclidean plane. We call graph $G = (V, E)$ to be *$\mathcal{M}$-intersection graph* if its vertices can be assigned to objects of $\mathcal{M}$ in such a way that any two vertices are adjacent if and only if two corresponding

objects have a non-empty intersection. The class of all $\mathcal{M}$-intersection graphs is denoted by $\mathsf{IG}(\mathcal{M})$.

In the rest of the chapter we introduce intersection graph classes that we use later. We also briefly mention their important properties.

Interval graphs. Probably the best studied class of graphs is interval graphs. Interval graphs are graphs representable by intersection graphs of intervals on a line. We can describe it as IG(intervals on line) using our previous notation.

There are several linear time algorithms [15, 37] which recognize interval graphs. Maximal clique, independent set and optimal coloring problems can be efficiently[1] solved for interval graphs.

Segment graphs. Natural extension of interval graphs are segment graphs. Segment graphs are defined as IG(straight line segments in the plane) and also commonly denoted as SEG.

The central point of segment graphs is so called *Scheinerman's conjecture* [64]:

Conjecture 1.3.1 (Scheinerman 1984)**.** *Every planar graph is the intersection graph of straight line segments in the plane.*

This conjecture brings lots of interest to SEG graphs. Scheinerman's conjecture was recently confirmed to be true by Chalopin and Gonçalves [20]. Another interesting result is due to Kratochvíl and Matoušek [45], who proved that there are segment graphs such that the description of any segment representation, specifying the coordinates of endpoints, may require an exponential number of bits. They also proved that recognition of SEG is NP-hard and is in PSPACE.

String graphs. One of the most general studied classes of intersection graphs is string graphs, which refers to intersection graphs of curves in the plane or IG("curves in plane").

It is an easy exercise to notice that all planar graphs are string graphs. It was proved by Chalopin, Gonçalves and Ochem [19] that every planar graph has a string representation in which each pair of strings has at most one crossing point, which was partial step towards Scheinerman's conjecture. Surprisingly, for a long time it was not known if recognition of STRING

[1] By *efficiency* we mean that there exists a polynomial time algorithm.

(that is another common way how to refer to string graphs) is decidable[2]. Decidability of recognition of string graphs was independently proven by Shaefer and Štefankovič [63] and by Pach and Tóth [56]. Kratochvíl and Matoušek [47] found graphs such that any string representation contains at least an exponential number of intersecting points. Therefore, it was very surprising when Shaefer, Sedgwick and Štefankovič [61] showed that recognition of string graphs is in NP.

There are many more intersection classes to mention. An interesting extension of interval graphs are graphs with *bounded boxicity*, and an extension of segment graphs are graphs of *convex regions*. We only mention here geometric intersection graphs.

[2]Problem is *decidable* if for its language there exists a finite time algorithm.

Chapter 2

Optimization problems of Segment Graphs

In this chapter we explore results devoted to the complexity of two classical optimization problems proposed to study by Kratochvíl and Nešetřil [48] which are related to segment graphs.

Problem:	MaxClique
Input:	a graph G and integer k
Output:	Is in G a clique of size at least k?

and

Problem:	MaxIndSet
Input:	a graph G and integer k
Output:	Is in G an independent set of size at least k?

Even approximations of these problems are known to be NP-hard for general graphs [5, 40]. On the other hand, the same problems are known to be polynomially solvable for interval graphs, and segment graphs as natural generalization of interval graphs may still hold some of these efficient properties.

Kratochvíl and Nešetřil [55] proved that MaxIndSet in 2-DIR is NP-hard, where 2-DIR are segment graphs which are representable by segments of only two distinct directions. On a more positive side is their other result:

Theorem 2.0.1. *The problem of* MaxClique *is polynomial time solvable when restricted to 2-DIR.*

The maximum clique problem for general segment graphs remained open. In the survey paper "On six problems posed by Jarik Nešetřil" by Bang-Jensen et al. [9] the general maximum clique problem was referred as being "among the most tantalizing unsolved problems in the area".

In 2005 Ambühl and Wagner [3] were studying MaxClique of ellipses in the plane and they proved the following theorem.

Theorem 2.0.2. *For every $\rho > 1$ the problem of* MaxClique *for intersection graphs of ellipses with aspect ratio ρ is NP-hard.*

You might notice that in case when ρ is exactly one, any such ellipse is a circle and in case ρ is infinity, then the ellipse is segment. Unfortunately, Theorem 2.0.2 was proven only for bounded ratios. There is another approach to show hardness of the problem. Rather than finding a reduction and showing that the problem is NP-hard one can find a class for which the problem is hard and is contained in the desired class. In this particular case, we want to show that every co-planar (complement of planar) graph is also a segment graph. If it is true then MaxClique would be NP-hard for co-planar graphs because MaxIndSet is NP-hard for planar graphs. In 1998, Kratochvíl and Kuběna [46] proved the following.

Theorem 2.0.3. *The co-planar graphs are intersection graphs of convex sets in plane.*

The next sections are devoted to our result, which is joint work with Mathew C. Francis and Jan Kratochvíl [27]. We show that the complements of partial-2-trees (which are defined in the next section) are segment graphs.

2.1 Partial 2-trees

We first define partial 2-trees.

Definition 2.1.1. *A* 2-tree *is K_2 or graph G having vertex v of degree 2 such that the neighbors of v are adjacent and $G - v$ is a 2-tree, see Fig. 2.1. A* partial 2-tree *is any subgraph of 2-tree.*

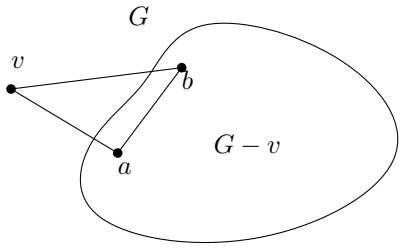


Figure 2.1: If $G - v$ is 2-tree then G is 2-tree.

We also define *spanning partial 2-tree* which is a spanning subgraph of a 2-tree. Notice, partial 2-trees form a subclass of planar graphs that includes series-parallel graphs and outerplanar graphs as proper subclasses. For further information on k-trees see [14].

2.2 Segments, rays and compatible segment representations

If a and b are two points in the plane, then:

- A *segment* with endpoints a and b is denoted as ab, is the set $\{a+\rho(b-a) : \rho \in [0,1]\}$. Any point on a segment that is not one of its endpoints is said to be an *interior* point of the segment. For the purposes of this section, we shall assume that every segment has a non-zero length—i.e., the endpoints of a segment may not coincide.

- A *ray* starting at a point a and passing through a point b is the set $\{a + \rho(b - a) : \rho \in [0, \infty)\}$. A ray has a single endpoint, which is its starting point.

Given l_1 and l_2, where each could be a segment or a ray, they are said to *cross* each other if $l_1 \cap l_2$ consists of one point that is not an endpoint of l_1 or l_2.

Given a segment graph G, there exists, by definition, a function $f : V(G) \to \mathcal{R}$ such that for distinct vertices u and v of G, u and v are adjacent if and only if $f(u)$ and $f(v)$ intersect, where $\mathcal{R}$ is a collection of segments. We say that $\mathcal{R}$ is a "segment representation" of G.

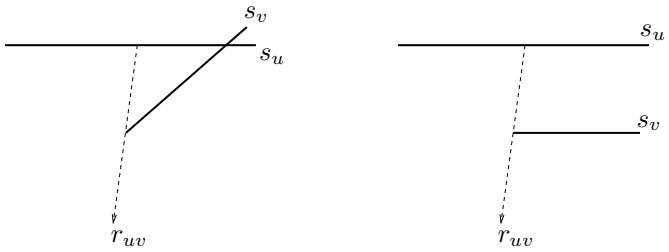


Figure 2.2: Segment representations for $\overline{G}$ that are compatible with G_T when $|V(G)| = 2$.

Definition 2.2.1. *Let G be a spanning subgraph of a 2-tree G_T. If $\overline{G}$ is a segment graph, then a segment representation $\mathcal{R}$ of $\overline{G}$ consisting of $\{s_u : u \in V(G)\}$ is* compatible with G_T *if for each $uv \in E(G_T)$, a ray r_{uv} can be drawn in $\mathcal{R}$ such that the collection of these rays satisfies the following properties:*

1. *r_{uv} starts from an interior point on one of s_u or s_v, passes through an endpoint of the other, and meets no other points of either s_u or s_v,*
2. *r_{uv} crosses every segment other than s_u and s_v, and*
3. *r_{uv} crosses every ray r_{xy} where $xy \in E(G_T) \setminus \{uv\}$.*

The rays r_{uv} where $uv \in E(G_T)$ shall be called "special rays".

2.3 The construction

Theorem 2.3.1. *If G is a spanning subgraph of a 2-tree G_T, then $\overline{G}$ has a segment representation that is compatible with G_T.*

Proof. We shall prove this by induction on $|V(G)|$. There is nothing to prove for $|V(G)| < 2$. If $|V(G)| = 2$, then $\overline{G}$ has a segment representation compatible with G_T as shown in Figure 2.2. Now let G be a spanning partial 2-tree G with $|V(G)| > 2$. By definition of a 2-tree, there exists a vertex v of degree 2 in G_T with neighbors x and y such that $xy \in E(G_T)$ and $G_T - v$ is also a 2-tree. Let $G' = G - v$; note that G' is also a spanning partial 2-tree, since it is a spanning subgraph of the 2-tree $G_T - v$. For ease of notation, we shall denote the 2-tree $G_T - v$ by G'_T. By our induction hypothesis, there is a segment representation for $\overline{G'}$ that is compatible with G'_T. We will extend

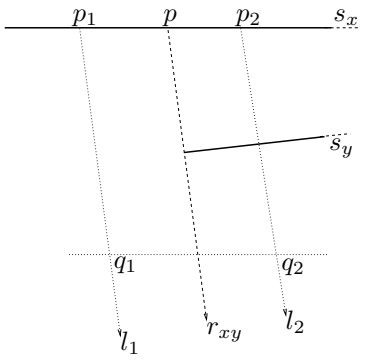


Figure 2.3: Starting point of construction

this to a segment representation for $\overline{G}$ that is compatible with G_T, thereby completing the proof.

Let $\mathcal{R}'$ be a segment representation of $\overline{G'}$ that is compatible with G'_T, and let s_u denote the segment corresponding to a vertex $u \in V(\overline{G'})$. We shall add a new segment s_v to $\mathcal{R}'$ to form the required segment representation $\mathcal{R}$ for $\overline{G}$.

Since $\mathcal{R}'$ is compatible with G'_T, there is a special ray r_{xy} in $\mathcal{R}'$, which we shall assume without loss of generality starts from an interior point of s_x and passes through an endpoint of s_y. Let p be the starting point of the ray r_{xy} on s_x (see Figure 2.3). Let l_1 and l_2 be rays starting from points p_1 and p_2 on s_x on either side of p and parallel to r_{xy} that cross every segment and special ray that r_{xy} crosses. By our definition of crossing, the ray r_{xy} meets every segment and special ray that it crosses at a point that is an endpoint of neither of them. Therefore, we can always choose rays l_1 and l_2 distinct from r_{xy} as long as s_x is not a point or parallel to r_{xy}. By our definition of r_{xy}, s_x cannot lie along r_{xy}, and also s_x cannot be a point; this ensures that the two rays l_1 and l_2 distinct from r_{xy} can be obtained. Of all the crossing points of segments and special rays on l_1, let q_1 be the farthest from p_1. Let q_2 be the similarly defined point on l_2. Observe that any segment or special ray that meets the segment p_1p_2 and crosses q_1q_2 crosses every segment and special ray that r_{xy} crosses.

The segment s_v is placed in the following way in $\mathcal{R}'$ to obtain $\mathcal{R}$:

Case 1. $yv \in E(G)$. We place s_v as shown in Figure 2.4. As $yv \notin E(\overline{G})$, the segment s_v is disjoint from the segment s_y. Let us first consider the case when $xv \notin E(G)$. In $\overline{G}$, v is adjacent to all the vertices in $V(G) \setminus \{v, y\}$. This

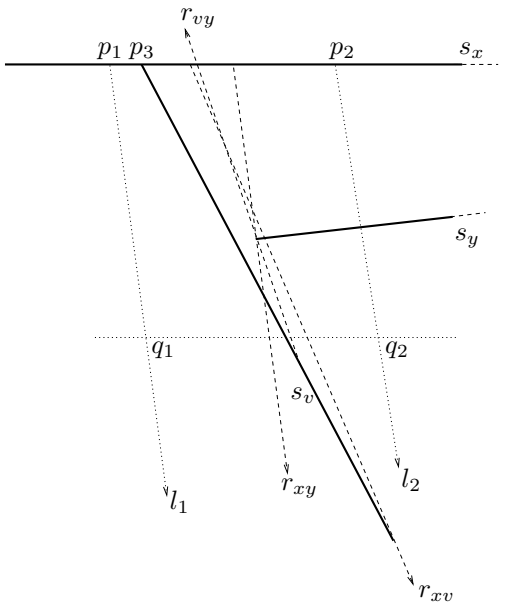


Figure 2.4: The case when $yv \in E(G)$

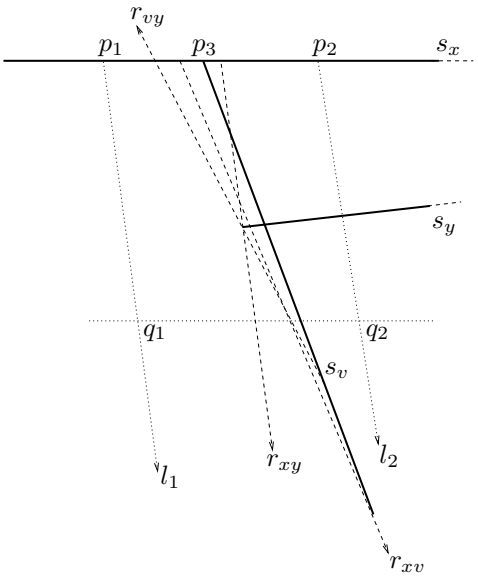


Figure 2.5: The case when $yv \notin E(G)$

requirement is satisfied, since s_v meeting p_1p_2 and crossing q_1q_2 implies that s_v crosses all the segments that cross r_{xy} and meets s_x too. Moreover, s_v also crosses all the special rays in $\mathcal{R}'$ including r_{xy}. We will show that we can now draw two rays r_{vy} and r_{xv}, so that they, together with the special rays in $\mathcal{R}'$, form the collection of special rays in $\mathcal{R}$ that make it compatible with G_T. The rays r_{vy} and r_{xv} can be drawn as shown in the figure so that they cross s_x and s_y respectively. In addition, both of them cross each other and r_{xy}. Since they also meet p_1p_2 and cross q_1q_2, each of them crosses all the special rays and segments that cross r_{xy}. Thus $\mathcal{R}$ is a segment representation of $\overline{G}$ that is compatible with G_T. Note that we can draw the segment s_v and the rays r_{vy} and r_{xv} in this way as long as the segments s_x and s_y have nonzero length and s_y does not lie along the ray r_{xy}. Our definitions of segments and special rays ensure that these pathological situations do not occur. Now, if $xv \in E(G)$, then s_v can be slightly shortened at the end p_3 so that it becomes disjoint from s_x without affecting any of the other arguments, so we still obtain the segment representation $\mathcal{R}$ for $\overline{G}$.

Case 2. $yv \notin E(G)$. We place s_v, r_{xv} and r_{vy} as shown in Figure 2.5. The rest of the argument is similar to that of Case 1.

We thus have a segment representation of $\overline{G}$ which is compatible with G_T. This completes the proof. □

Corollary 2.3.2. *If G is any partial 2-tree, then $\overline{G}$ is a segment graph.*

Proof. Let G' be the spanning partial 2-tree obtained by adding the required number of isolated vertices to G. By Theorem 2.3.1, $\overline{G'}$ has a segment representation, say $\mathcal{R}$. As $\overline{G}$ is an induced subgraph of $\overline{G'}$, the segments corresponding to the vertices in $V(G') \setminus V(G)$ can be removed from $\mathcal{R}$ to obtain a segment representation of $\overline{G}$. □

Because series-parallel graphs and outerplanar graphs are subclass of partial 2-trees the following holds.

Corollary 2.3.3. *The complements of series-parallel graphs are segment graphs.*

Corollary 2.3.4. *The complements of outerplanar graphs are segment graphs.*

Chapter 3

Intersection Graphs of Paths

In this chapter we study another class of intersection graphs: **V**ertex-intersection graphs of **P**aths in **G**rids[1] (VPG graphs) introduced by Asinowski et. al [7, 8].

Definition 3.0.1. *A* VPG-representation *of a graph G is a collection of paths of the rectangular grid where the paths represent the vertices of G in such a way that two vertices of G are adjacent if and only if the corresponding paths share at least one vertex.*

VPG-representations arise naturally when studying circuit layout problems and layout optimization where layouts are modeled as paths (wires) on grids. For more details see [65]. One approach to minimize the cost or difficulty of production involves minimizing the number of times the wires bend [16, 54]. Therefore, we focus here on VPG-representations parameterized by the number of times each path is allowed to *bend* (these representations are also the focus of [7, 8]). In particular, a *k-bend path* is a path in the grid which contains at most k *bends* where a *bend* is when two consecutive edges on the path have different horizontal/vertical orientation. Analogically to VPG-representation we define the following.

Definition 3.0.2. *B_k-VPG representation of graph G is a VPG-representation of G where each path is a k-bend path. A graph is a B_k-VPG graph if it has a B_k-VPG representation.*

Several relationships between VPG graphs and some well-known graph classes (i.e., circle graphs, circular arc graphs, interval graphs, planar graphs,

[1]The grids to which we refer are always rectangular.

segment graphs, and string graphs) were observed by Asinowski et al. [7, 8]. In particular, they showed that every planar graph is a B_3-VPG graph. They also conjectured that planar graphs are not contained in B_2-VPG, which turned out to be *false* due to the result of Chaplick and Ueckerdt [21], who proved the following.

Theorem 3.0.5. *Every planar graph is a B_2-VPG graph.*

This result was recently strengthen by Biedl and Derk [12], who proved that every planar graph has B_2-VPG representation where representations of every two vertices cross at most once[2].

In [7] one can find a proof of a folklore result.

Theorem 3.0.6. *VPG = STRING*

Consequently, we can apply several results for STRING to VPG. The recognition problem for the VPG (= STRING) is known to be NP-hard by Kratochvíl [43] and is contained in NP by a beautiful and surprising result of Schaefer, Sedgwick, and Štefankovič [62].

The base case B_0-VPG is a special case of SEG-graphs more well-known as 2-DIR[3]. It is NP-complete to recognize 2-DIR (= B_0-VPG graphs) [44]. However, the recognition status of B_k-VPG for every $k > 0$ was cojectured in [8] (all cases were conjectured to be NP-complete). In the next sections we confirm this conjecture by proving a stronger result.

Furthermore, in [7, 8] it is shown that B_0-VPG $\subsetneq$ B_1-VPG $\subsetneq$ VPG and it was conjectured that B_k-VPG $\subsetneq$ B_{k+1}-VPG for every $k > 0$. We confirm this conjecture constructively, see Section 3.2.

3.1 Noodle-Forcing Lemma

In this section, we present the key lemma (see Lemma 3.1.1). Essentially, we prove that, for "proper" representations R of a graph G, there is a graph G' where G is an induced subgraph of G' and R is "sub-representation" of every representation of G' (i.e., all representations of G' require the part corresponding to G to have the "topological structure" of R). We begin this section with several definitions.

[2]This property is also called 1-STRING.

[3]the intersection graphs of straight line segments in the plane with at most two distinct directions (slopes)

Let $G = (V, E)$ be a graph. A *representation* of G is a collection $R = \{R(v),\, v \in V\}$ of piecewise linear curves in the plane, such that $R(u) \cap R(v)$ is nonempty iff uv is an edge of G.

An *intersection point* of a representation R is a point in the plane that belongs to (at least) two distinct curves of R. Let $\mathrm{In}(R)$ denote the set of intersection points of R.

A representation is *proper* if

1. each $R(v)$ is a simple curve, i.e., it does not intersect itself,
2. R has only finitely many intersection points (in particular no two curves may overlap) and finitely many bends, and
3. each intersection point p belongs to exactly two curves of R, and the two curves cross in p (in particular, the curves may not touch, and an endpoint of a curve may not belong to another curve).

Let R be a proper representation of $G = (V, E)$, let R' be another (not necessarily proper) representation of G, and let ϕ be a mapping from $\mathrm{In}(R)$ to $\mathrm{In}(R')$. We say that ϕ is *order-preserving* if it is injective and has the property that for every $v \in V$, if $p_1, p_2, \ldots, p_k$ are all the distinct intersection points on $R(v)$, then $\phi(p_1), \ldots, \phi(p_k)$ 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 $R(v)$. (If $R'(v)$ visits the point $\phi(p_i)$ more than once, we may select one visit of each $\phi(p_i)$, such that the selected visits occur in the correct order $\phi(p_1), \ldots, \phi(p_k)$.)

For a set P of points in the plane, the *ε-neighborhood of P*, denoted by $\mathcal{N}_\varepsilon(P)$, is the set of points that have distance less than ε from P.

Lemma 3.1.1 (Noodle-Forcing Lemma)**.** *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 every $v \in V$, and $R'(w)$ is a horizontal or vertical 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 by 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 ε-neighborhood of $R(v)$, and $R(v)$ is contained in the ε-neighborhood of $R^\varepsilon(v)$.*

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

Proof. Suppose we are given a proper representation R of a graph G. We say that a point in the plane is *a special point* of R, if it is an endpoint of a curve in R, a bend of a curve in R, or an intersection point of R.

Before we describe the graph G', we first construct an auxiliary graph H which is a subdivision of a 3-connected plane graph whose drawing overlays the representation R and has the following properties.

P1 The edges of H are drawn as vertical and horizontal segments, and every internal face of H is a rectangle (possibly containing more than four vertices of H on its boundary). 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 f.

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

We construct the plane graph H in several steps. In the first step, we produce a square grid H_0 such that the whole representation R is contained in the interior of H_0, i.e., no part of R intersects the outer face of H_0. We may further assume that the grid H_0 is fine enough so that it has the properties P1–P3 above. See Fig. 3.1.

If the graph H_0 does not satisfy all the properties P1–P6, we will further subdivide some of the faces of H_0. Suppose first that H_0 has an edge e that is intersected more than once by the curves of R (see Fig. 3.2). We then add a new edge e' into the drawing of H_0, which is parallel to e and embedded very close to e, thus splitting a face adjacent to e into two new faces. We

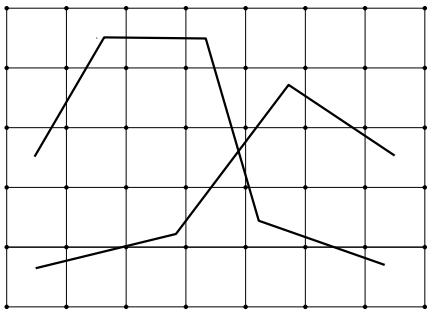


Figure 3.1: An embedding of a graph H_0 over a representation R.

then split the face incident to e and e' by new edges perpendicular to e and e', in such a way that each intersection point of e or e' with a curve of R belongs to a different edge in the new graph. Performing this operation for every edge of H_0, we obtain a new graph H_1, which satisfies the properties P1–P4.

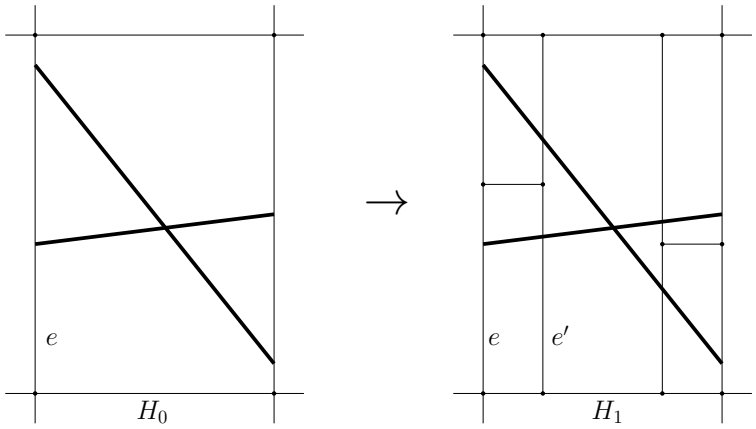


Figure 3.2: An embedding of a graph H_1 over a representation R.

If the graph H_1 fails to satisfy P5 or P6, it means that H_1 has a face f whose intersection with R contains two disjoint curves p and q, each of which is a subcurve of a curve from R (see Fig. 3.3). In this case, we draw in f a piecewise linear curve c with horizontal and vertical segments which cuts f into two pieces such that one contains p and the other q. We then extend the segments of c to form a grid-like subdivision of the face f into subfaces, each of which is only intersected by at most one of p and q. If necessary, we may

further subdivide the newly created faces to make sure they do not violate P4. In this way, we obtain the graph H satisfying all the properties P1–P6.

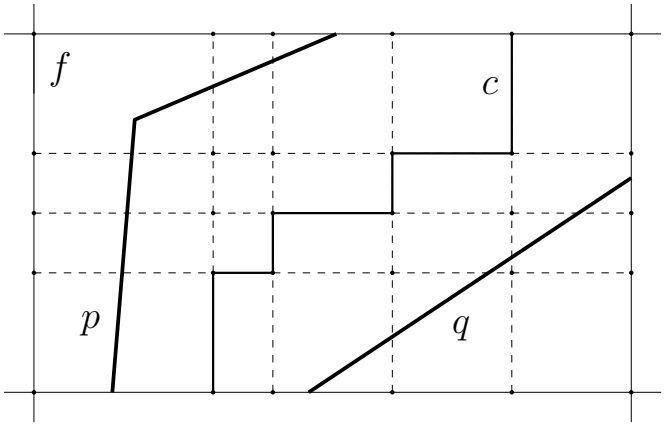


Figure 3.3: A description of the transformation of H_1 to H_2.

We now transform H into an arrangement S of horizontal and vertical segments. See Figure 3.4. For every vertex $v \in V(H)$, the set S contains two segments $S_1(v)$ and $S_2(v)$ which intersect close to the point v. We call these two segments the *vertex-segments of* v. We assume that the vertex segments do not intersect any of the curves of R, and they do not overlap with any edge of H. For every edge $e \in E(H)$, we further put into S a segment $S(e)$, called the *edge-segment of* e, which partially overlaps with e, intersects exactly those curves of R that e intersects, and does not intersect any vertex-segment or any other edge-segment of S. Finally, for any vertex v that is incident to an edge e of H, we put into S a segment $S(v, e)$ that intersects both $S(e)$ and the vertex segment of v that is parallel to $S(e)$, and does not intersect any other segment of S or curve of R. We call $S(v, e)$ *the connector of* v *and* e.

Define a representation $R' = R \cup S$, and let G' be the graph determined by this representation. Clearly, G is an induced subgraph of G', and the representation R' has the properties stated in the lemma. It remains to show that every representation of G' can be transformed into a representation R^ε that has the two properties stated in the lemma. We will first show that every representation of G' can be transformed into a representation that satisfies the first property, and then we will show that the second property follows from the first one.

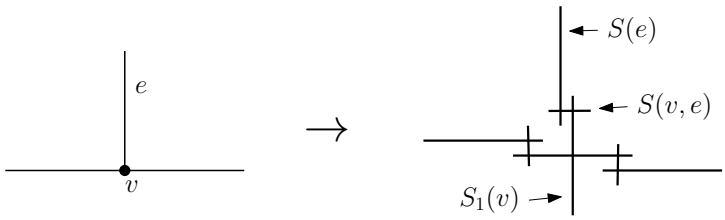


Figure 3.4: Transforming the graph H into a segment arrangement S.

Consider an arbitrary representation R'' of G'. The curves in R'' that correspond to the vertex-segments, edge-segments and connectors will be referred to as vertex-curves, edge-curves and connector curves. Let $S''(e)$ be an edge-curve that represents an edge $e \in E(H)$ in R''. Let c_1 and c_2 be its two adjacent connector curves. Let $\gamma(e)$ be a minimal subcurve of $S''(e)$ with the property that its two endpoints belong respectively to c_1 and c_2. In particular, $\gamma(e)$ has no other intersections with the connector curves apart from its endpoints. Call $\gamma(e)$ *the main part* of $S''(e)$. Next, for each connector curve, consider its minimal subcurve whose one endpoint belongs to the main part of its adjacent edge-curve, and the other endpoint belongs to its adjacent vertex-curve. Call this subcurve *the main part* of the connector curve.

If, in the representation R'', we contract each intersecting pair of vertex-curves into a single point, and then replace each connector curve and edge-curve by their main parts, we obtain a drawing of the graph H, where a vertex is represented by the contracted pair of vertex-curves, and an edge corresponds to the main part of its edge-curve together with the adjacent connectors. Since H is a subdivision of a 3-connected graph, each of its drawings can be transformed into any other drawing by a circular inversion and a homeomorphism. In particular, we may transform R'' in such a way that the main part of any edge-curve is contained inside the corresponding edge-segment $S(e)$, the main part of any connector curve is contained inside the corresponding connector, and every vertex-curve representing a vertex $v \in V(H)$ is embedded in a small neighborhood of $S_1(v) \cup S_2(v)$. Suppose therefore that R'' has been transformed in this way.

Let f be an internal face of H, whose boundary is formed by k edges $e_1, \ldots, e_k$ and k vertices $v_1, \ldots, v_k$. Consider the union of the main parts $\gamma(e_1), \ldots, \gamma(e_k)$, together with the main parts of their adjacent connectors and together with the vertex curves representing the vertices $v_1, \ldots, v_k$. In

the union of these curves, there is a unique bounded region that is adjacent to all of the curves $\gamma(e_1), \ldots, \gamma(e_k)$. We call this region the *pseudo-face* corresponding to f, denoted by $\bar{f}$. Notice that an edge-curve $R''(e)$ may only intersect the two pseudo-faces that correspond to the two faces adjacent to e in H.

Let v be a vertex of G. Let $f_0, f_1, \ldots, f_k$ be the faces of H intersected by the curve $R(v)$ in the order from one endpoint to the other. Let $e_1, \ldots, e_k$ be the edges of H crossed by $R(v)$, where e_i is adjacent to f_{i-1} and f_i. Notice that we have $k \geq 2$ due to the property P3.

Since the curve $R''(v)$ must intersect the edge-curve $S''(e_i)$ for any $i = 1, \ldots, k$, it must intersect at least one of the two pseudo-faces $\bar{f}_{i-1}$ and $\bar{f}_i$. Also, since $R''(v)$ does not intersect any edge-curve other than the k curves $S''(e_i)$ for $i = 1, \ldots, k$, it cannot enter into any pseudo-face other than $\bar{f}_i$ for $i = 0, \ldots, k$.

Note, however, that $R''(v)$ does not necessarily intersect the main part of $S''(e_1)$ and that it does not necessarily penetrate into $\bar{f}_0$, and similarly for $S''(e_k)$ and $\bar{f}_k$ (see Fig. 3.5). Overall, we see that $R''(v)$ enters a pseudo-face $\bar{f}$ if and only if $R(v)$ enters the face f, except possibly for the two faces f_0 and f_k. We also see that $R''(v)$ crosses the main part of an edge-curve $S''(e)$ if and only if $R(v)$ crosses e, except possibly for the two edges e_1 and e_k. From this it is easy to see that for any $\varepsilon > 0$, we may deform $R''(v)$ so that it will belong to $\mathcal{N}_\varepsilon(R(v))$, without changing the boundary of any pseudo-face. Moreover, by possibly creating a 'bulge' in $\gamma(e_0)$ and $\gamma(e_k)$ as shown in Fig. 3.5, we may ensure that every point of $R(v)$ has a point of $R''(v)$ in its ε-neighborhood. This shows that R'' is homeomorphic to a representation R^ε having the first property stated in the lemma.

We will now show that if ε is small enough, then every representation R^ε that satisfies the first property in the statement of the lemma must admit an order-preserving mapping $\phi\colon \mathrm{In}(R) \to \mathrm{In}(R^\varepsilon)$.

Let $N(u)$ denote the set $\mathcal{N}_\varepsilon(R(u))$. We call $N(u)$ *the noodle* of u. Suppose that two curves $R(u)$ and $R(v)$ have k mutual crossing points $p_1, \ldots, p_k$. If ε is small enough, the intersection $N(u) \cap N(v)$ has k connected components $P_1, \ldots, P_k$, each of them being an open parallelogram, and each P_i containing a unique intersection point p_i. We call P_i *the zone of* p_i. By making ε small, we may assume that the zones of the points in $\mathrm{In}(R)$ are disjoint, that every noodle has a boundary that is a simple closed curve, and that the boundary of a noodle does not intersect any of the zones.

Consider now the curve $R^\varepsilon(u)$ for some vertex $u \in V$. We may assume

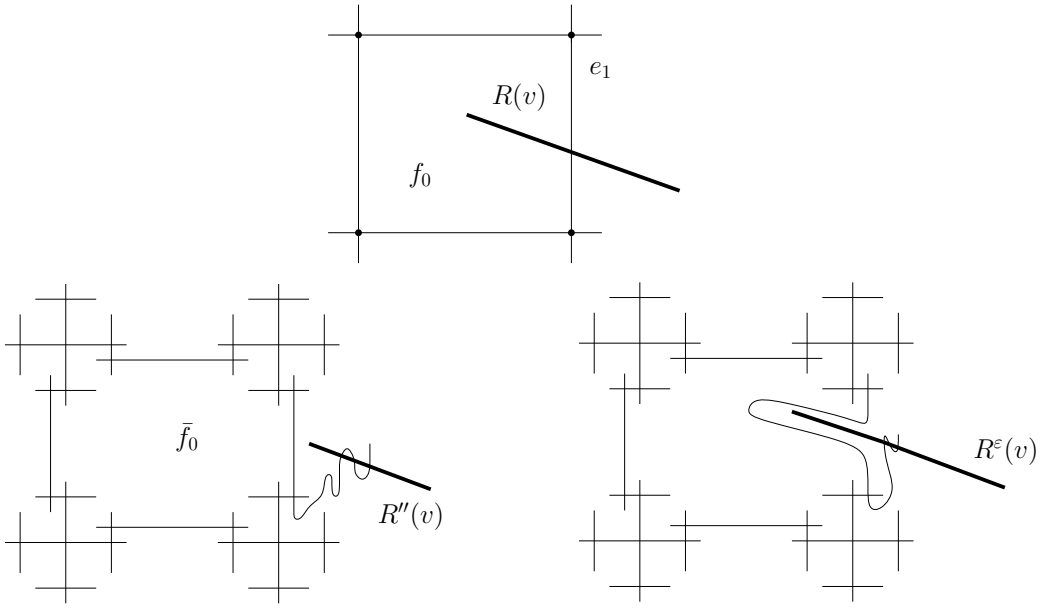


Figure 3.5: A situation where $R''(v)$ does not enter the pseudo-face $\bar{f}_0$.

that the endpoints of $R^\varepsilon(u)$ coincide with the endpoints of $R(u)$ (otherwise we may shorten $R^\varepsilon(u)$ and deform it in a neighborhood of the endpoints). Choose an orientation of $R(u)$ (i.e., choose an initial endpoint), and suppose that $R^\varepsilon(u)$ has the same orientation. Suppose that the curve $R(u)$ has m crossing points $q_1, \ldots, q_m$, encountered in this order. Let $Q_1, \ldots, Q_m$ be their zones. Fix a zone Q_i and consider the intersection $R^\varepsilon(u) \cap Q_i$. This intersection is a union of subcurves of $R^\varepsilon(u)$. Choose such a subcurve whose endpoints lie on the opposite sides of Q_i; if there are more such subcurves, choose the first such subcurve visited when $R^\varepsilon(u)$ is traced in the direction of its orientation. Call the chosen subcurve *the representative of* $R^\varepsilon(u)$ *in* Q_i, denoted by $r_i(u)$. Note that when $R^\varepsilon(u)$ is traced from beginning to end, the representatives are visited in the order $r_1(u), r_2(u), \ldots, r_m(u)$.

We now define the order-preserving mapping ϕ. See Fig. 3.6. Let $p \in \mathrm{In}(R)$ be an intersection of two curves $R(u)$ and $R(v)$, and let P be the zone of p. Let $r(u)$ and $r(v)$ be the representatives of $R(u)$ and $R(v)$ inside P. Define $\phi(p)$ to be an arbitrary intersection of $r(u)$ and $r(v)$. We may apply a homeomorphism inside P to make sure that $\phi(p)$ coincides with p. By the choice of representatives, ϕ is order-preserving. This proves the lemma. □

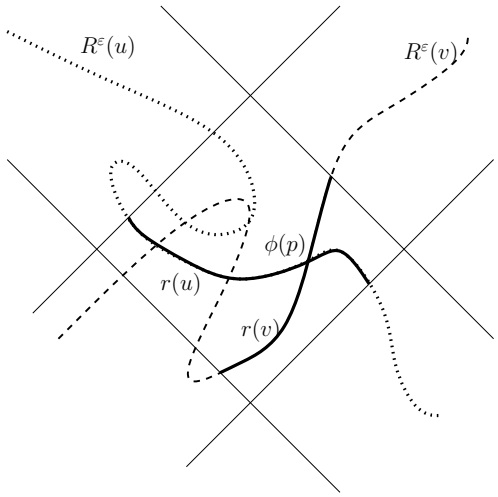


Figure 3.6: Finding a crossing $\phi(p)$ inside the zone of p. The fat parts of the curves are the representatives.

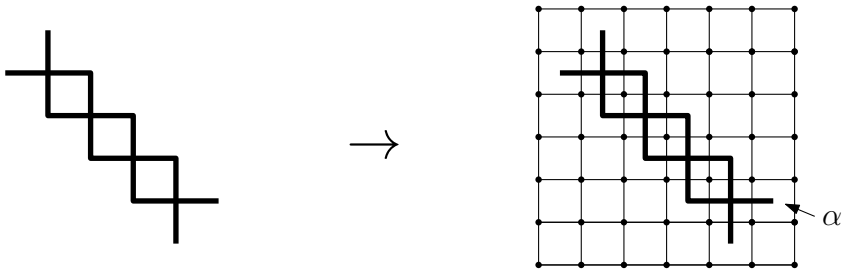


Figure 3.7: The sausage representation for $k = 3$.

3.2 Relations between classes

With the Noodle-Forcing Lemma, we can prove our separation results.

Theorem 3.2.1. *For any $k \geq 1$, there is a graph G' that has a proper representation using k-bend axis-parallel curves, but has no representation using $(k-1)$-bend axis-parallel curves.*

Proof. Consider the graph K_2 consisting of a single edge uv, with a representation R in which both u and v are represented by weakly increasing k-bend staircase curves that have $k+1$ common intersections $p_1, \ldots, p_{k+1}$, in left-to-right order, see Fig.3.7.

Apply the Noodle-Forcing Lemma to K_2 and R, to obtain a graph G' with a k-bend representation R'. We claim that G' has no $(k-1)$-bend representation. Assume for contradiction that there is a $(k-1)$-bend representation R'' of G'. Lemma 3.1.1 then implies that there is an order-preserving mapping $\phi\colon \mathrm{In}(R) \to \mathrm{In}(R'')$. Let $s_i(u)$ be the subcurve of $R''(u)$ between the points $\phi(p_i)$ and $\phi(p_{i+1})$, and similarly for $s_i(v)$ and $R''(v)$. Consider, for each $i = 1, \ldots, k$, the union $c_i = s_i(u) \cup s_i(v)$. We know from Lemma 3.1.1 that $s_i(u)$ and $s_i(v)$ cannot completely overlap, and therefore the closed curve c_i must surround at least one nonempty bounded region of the plane. Therefore c_i contains at least two bends different from $\phi(p_i)$ and $\phi(p_{i+1})$. We conclude that $R''(u)$ and $R''(v)$ together have at least $2k$ bends, a contradiction. □

Straight from this lemma we get as a consequence the following.

Corollary 3.2.2. *B_k-VPG $\subsetneq$ B_{k+1}-VPG*

Because two straight-line segments in the plane cross at most once, the Noodle-Forcing Lemma also implies following.

Corollary 3.2.3. *B_k-VPG $\not\subset$ SEG, for any $k \geq 1$.*

Is there some k such that every SEG graph is contained in B_k-VPG? The following lemma answers this question.

Theorem 3.2.4. *For every k, there is graph which belongs to 3-DIR but not to B_k-VPG.*

Proof. We fix an arbitrary k. Consider, for an integer n, a representation $R \equiv R(n)$ formed by $3n$ segments, where n of them are horizontal, n are vertical and n have a slope of 45 degrees. Suppose that every two segments of R with different slopes intersect, and their intersections form the regular pattern depicted in Figure 3.8 (with a little bit of creative fantasy this pattern resembles a waffle).

Note that the representation R forms $\Omega(n^2)$ empty internal triangular faces bounded by segments of R, and the boundaries of these faces intersect in at most a single point. Suppose that n is large enough, so that there are more than $3kn$ such triangular faces. Let G be the graph represented by R.

The representation R is proper, so we can apply the Noodle-Forcing Lemma to R and G, obtaining a graph G' together with its 3-DIR representation R'. We claim that G' has no B_k-VPG representation.

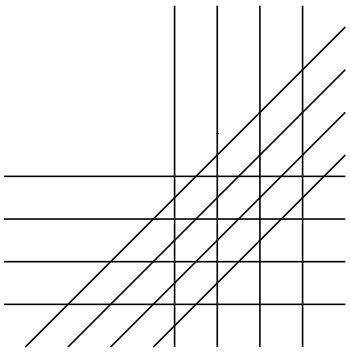


Figure 3.8: The 'waffle' representation R from Lemma 3.2.4.

Suppose for contradiction that there is a B_k-VPG representation R'' of G'. We will show that the $3n$ curves of R'' that represent the vertices of G contain together more than $3kn$ bends.

From the Noodle-Forcing Lemma, we deduce that there exists an order-preserving mapping $\phi\colon \mathrm{In}(R_n) \to \mathrm{In}(R''_n)$. Let T be a triangular face of the representation R. The boundary of T consists of three intersection points $p, q, r \in \mathrm{In}(R)$ and three subcurves a, b, c. The three intersection points $\phi(p)$, $\phi(q)$ and $\phi(r)$ determine the corresponding subcurves a'', b'' and c'' in R''.

The Noodle-Forcing Lemma implies that there is a homeomorphism h which sends a'', b'', and c'' into small neighborhoods of a, b and c, respectively. This shows that each of the three curves a'', b'' and c'' contains a point that does not belong to any of the other two curves. This in turn shows that at least one of the three curves is not a segment, i.e., it has a bend in its interior.

Since the triangular faces of R have non-overlapping boundaries, and since ϕ is order-preserving, we see that for each triangular face of R there is at least one bend in R'' belonging to a curve representing a vertex of G. Since G has $3n$ vertices and R determines more than $3kn$ triangular faces, we conclude that at least one curve of R'' has more than k bends, a contradiction. □

3.3 Hardness Results

In this section we strengthen the separation result of Corollary 3.2.2 by showing that not only are the classes B_k-VPG and B_{k+1}-VPG different, but providing a B_{k+1}-VPG representation does not help in deciding B_k-VPG mem-

bership. This also settles the conjecture on NP-hardness of recognition of these classes stated in [8], in a considerably stronger form than it was asked.

Theorem 3.3.1. *For every $k \geq 0$, deciding membership in B_k-VPG is NP-complete even if the input graph is given with a B_{k+1}-VPG representation.*

Proof. We use the NP-hardness reduction developed in [44] for showing that recognizing grid intersection graphs is NP-complete. Grid intersection graphs are intersection graphs of vertical and horizontal segments in the plane with additional restriction that no two segments of the same direction share a common point. Thus these graphs are formally close but not equal to B_0-VPG graphs[4]. However, bipartite B_0-VPG graphs are exactly grid intersection graphs. This follows from a result of Bellantoni et al. [10] who proved that bipartite intersection graphs of axes parallel rectangles are exactly grid intersection graphs.

The reduction in [44] constructs, given a Boolean formula Φ, a graph G_Φ which is a grid intersection graph if and only if Φ is satisfiable. In arguing about this, a representation by vertical and horizontal segments is described for a general layout of G_Φ for which it is also shown how to represent its parts corresponding to the clauses of the formula, referred to as *clause gadgets*, if at least one literal is true. The clause gadget is reprinted with a generous approval of the author in Fig. 3.9, while Fig. 3.10 shows the grid intersection representations of satisfied clauses, and Fig. 3.11a shows the problem when all literals are false. In Fig. 3.11b, we show that in the case of all false literals,

[4]where paths of the same direction are allowed to overlap

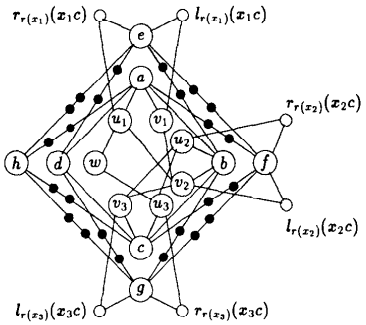


Figure 3.9: The clause gadget

the clause gadget can be represented by grid paths with at most 1 bend each. It follows that $G_\Phi \in B_1$-VPG and a 1-bend representation can be constructed in polynomial time. Thus, recognition of B_0-VPG is NP-complete even if the input graph is given with a B_1-VPG representation.

We use a similar approach for arbitrary $k > 0$ with a help of the Noodle-Forcing Lemma. We grill the same representation R of K_2 as in the proof of Theorem 3.2.1. We call the resulting graph $P(u)$ where u is one of the vertices of the K_2, the one whose curve in R is ending in a boundary cell denoted by α in the schematic Fig. 3.7. We call this graph the *pin* since it follows from Lemma 3.1.1 that it has a B_k-VPG representation such that the bounding paths of the cell α wrap around the grill and the last segment of $R(u)$ extends arbitrarily far (see the schematic Fig. 3.12). We will refer to this extending segment as the *tip* of the pin. It is crucial to observe that in any B_k-VPG representation R' of $P(u)$ all bends of $R'(u)$ are consumed between the crossing points with the curve representing the other vertex of K_2 and hence the part of $R'(u)$ that lies in the α cell of R' is necessarily straight.

Next we combine two pins together to form a *clothespin*. The construction is illustrated in the schemtic Fig. 3.13. We start with a K_4 whose edges are subdivided by one vertex each. Every STRING representation of this graph contains 4 basic regions which correspond to the faces of a drawing of the K_4 (this is true for every 3-connected planar graph and it is seen by contracting the curves corresponding to the degree 2 vertices, the argument going back to Sinden [65]). We add two vertices x_1, x_2 that are connected by paths of length 2 to the boundary vertices of two triangles, say β_1 and β_2. The curves representing x_1 and x_2 must lie entirely inside the corresponding regions.

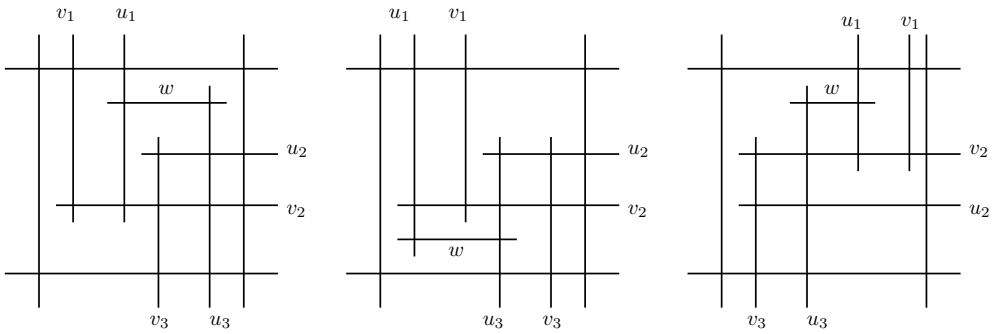


Figure 3.10: The representations of satisfied clauses.

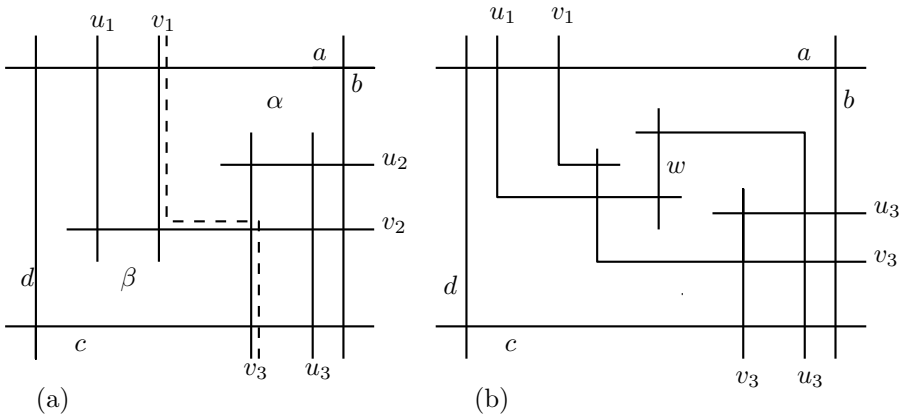


Figure 3.11: (a) The problem preventing the representation of an unsatisfied clause. (b) The representation of an unsatisfied clause gadget via curves with one bend.

Then we add two pins, say $P(u_1)$ and $P(u_2)$, connect the vertices of the boundary of α_i to x_i by paths of length 2 and make u_i adjacent to a vertex on the boundary of (for $i = 1, 2$). Finally, we add a third pin $P(u_3)$ and make u_3 adjacent to u_1 and u_2. We denote the resulting graph by $CP(u)$.

It is easy to check that the clothespin has a B_k-VPG representation $\check{R}$ such that the tips of $\check{R}(u_1)$ and $\check{R}(u_2)$ are parallel and extend arbitrarily far from the rest of the representation, as indicated in Fig. 3.13.

On the other hand, in any B_k-VPG representation R' of $CP(u)$, if a curve crosses $R'(u_1)$ and $R'(u_2)$ and no other path of $R'(CP(u))$, then it must cross the tips of $R'(u_1)$ and $R'(u_2)$. This follows from the fact that for $i = 1, 2$, $R'(x_i)$ must lie in α_i (to be able to reach all its bounding curves), and hence,

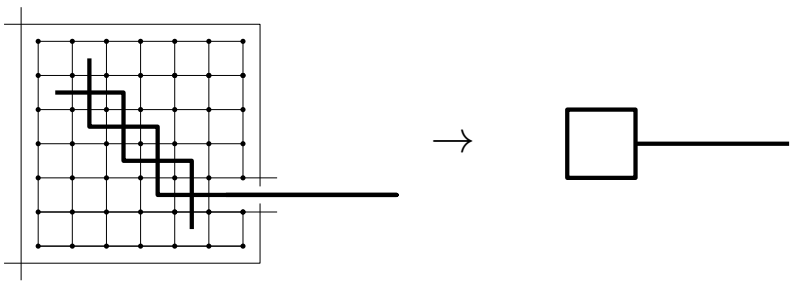


Figure 3.12: Construction of a pin

by circle inversion, all bends of $R'(u_i)$ are trapped inside β_i. If a curve crosses both $R'(u_1)$ and $R'(u_2)$, it must cross them outside $\beta_1 \cup \beta_2$, and hence it only may cross their tips.

Now we are ready to describe the construction of G'_Φ. We take G_Φ as constructed in [44] replace every vertex u by a clothespin $CP(u)$, and whenever $uv \in E(G_\Phi)$, we add edges $u_i v_j, i, j = 1, 2$. Now we claim that $G'_\Phi \in B_k$-VPG if and only if Φ is satisfiable, while $G'_\Phi \in B_{k+1}$-VPG is always true.

On one hand, if $G'_\Phi \in B_k$-VPG and R' is a B_k-VPG representation of G'_Φ, then the tips of $R'(u_1), u \in V(G_\Phi)$ form a 2-DIR representation of G_Φ ($R'(u_1)$ and $R'(v_1)$ may only intersect in their tips) and Φ is satisfiable.

On the other hand, if Φ is satisfiable, we represent G_Φ as a grid intersection graph and replace every segment of the representation by a clothespin with slim parallel tips and the body of the pin tiny enough so that does not intersect anything else in the representation. Similarly, if Φ is not satisfiable, we modify a 1-bend representation of G_Φ by replacing the paths of the representation by clothespins with 1-bend on the tips, thus obtaining a B_{k+1}-VPG representation of G_Φ. The representation consists of a large part inherited from the representation of G_Φ and tiny parts representing the heads of the pins, but these can be made all of the same constant size and thus providing only a constant ratio refinement of the representation of G_Φ. The representation is thus still of linear size and can be constructed in polynomial time. □

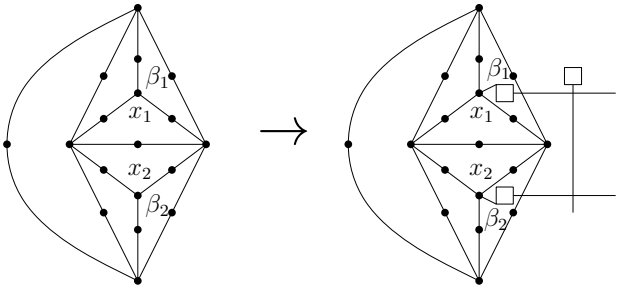


Figure 3.13: Construction of a clothespin

Chapter 4

Intersection Graphs of Islands

In this chapter we introduce the class of islands in extended grid which is closely related to VPG class of graphs and string graphs. We start here with necessary definitions.

Extended Grid. Informally speaking, the *extended grid* is the square grid with added diagonal edges, see Fig. 4.1. Formally speaking, the vertices of extended grid are the points of plane with integral coordinates.

Two vertices u and v defined by coordinates (X_u, Y_u) and (X_v, Y_v) are *adjacent* if and only if $|X_u - X_v| \leq 1$ and $|Y_u - Y_v| \leq 1$. Let u and v be adjacent then we call edge uv *horizontal* if $Y_u = Y_v$ (respectively, vertical if $X_u = X_y$) and *diagonal* if $|X_u - X_v| = 1$ and $|Y_u - Y_v| = 1$. Notice, it differs from the square grid only by diagonal edges. In the rest of the chapter we refer to the extended grid just as the grid.

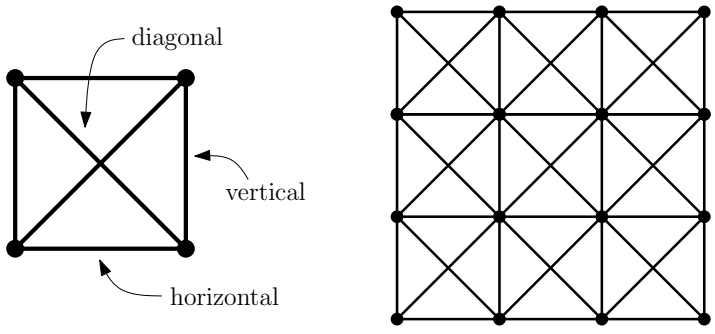


Figure 4.1: On the right we show horizontal, vertical and diagonal edges. On the left we show how a part of extended grid looks like.

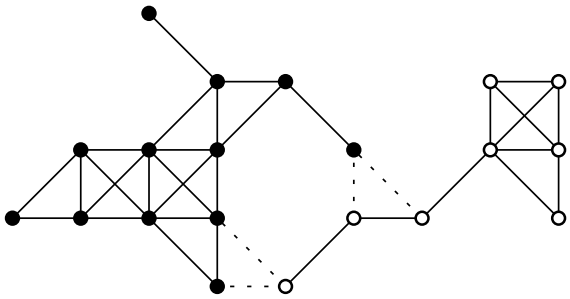


Figure 4.2: Two islands, one with black vertices and one with white vertices, are adjacent (dotted).

Island. An *island* is a non-empty set of grid vertices which induces a connected subgraph. We say that two islands U and V are *adjacent* if $U \cup V$ induces a connected subgraph, or equivalently, if there exist vertices $u \in U$ and $v \in V$ such that u and v are adjacent. We refer to a graph induced by island I as I_g. Island I is called k-island if $|I| \leq k$, that is, the number of vertices (of the grid) of I is less than or equal to k.

Faithful Representation. A faithful representation of graph G by islands is set $\mathcal{I}$ of vertex disjoint islands in the extended grid, such that the graph of $\mathcal{I}$ is isomorphic to G. The class of graphs which has faithful representation by islands will be denoted by ISLAND.

Graph classes. For a given set of k-islands $\mathcal{I}$, the relation of being adjacent (on islands) defines a graph. All graphs with such a property are called k-island graphs and define a graph class k-ISLAND. If we drop the condition which bounds the size of islands, we get ISLAND graph class.

4.1 Motivation from adiabatic quantum computing

Our problems are motivated by the proposed *adiabatic quantum computer* AQC [24]. AQC allows a physical system to evolve from an initial state to a final state following the principles of adiabatic evolution; by contrast the standard model of quantum computing involves the application of a sequence of gates to achieve this evolution. (It was shown in [1] that the two models are computationally equivalent.) The qubits and couplers of an adiabatic

quantum computer can be naturally seen to form a graph. This abstraction away from the underlying physics allows us to translate computational problems into a graph theoretic language. An AQC depends upon its hardware to find a binary vector that optimizes some objective function. In one model, to goal is to maximize an unconstrained quadratic objective function

$$f(x) = \sum_{i=1}^{n} h_i x_i + \lambda \sum_{i=1}^{n} \sum_{j=i+1}^{n} J_{ij} x_i x_j,$$

where λ is a (Lagrange) multiplier, h is a vector, and J is an upper-triangular matrix, describing respectively the linear and quadratic objective functions. For instance, by taking for h the all-one vector and for J the (upper half of the) adjacency matrix of G, and setting λ to be smaller than -2, we can formulate the maximum independent set problem in G.

One specific AQC is layed out like a finite portion of the extended grid. Each qubit in the hardware is a vertex in the extended grid and each coupler an edge [4]. It may seem that only a limited set of problems can be posed in this specific graph (the extended grid). However, it is possible to constrain certain qubits to take on the same spin, and in this way to create *islands*, i.e., connected subgraphs (or equivalently subtrees) of the extended grid, which can be treated as 'supervertices'. To represent an arbitrary graph G then amounts to assigning to each vertex of G an island of the extended grid, and to each edge of G an edge connecting the corresponding islands. Note that adjacent vertices of G must be assigned to islands that are sufficiently near to be connected by a bridge. If we additionally want to ensure that no noise or information in the quantum system transfer between islands through inactive couplers, we further require that nonadjacent vertices of G be assigned islands that are not sufficiently near to be connected by an edge of the extended grid. In such a case, we seek a faithful representation of G as defined earlier.

4.2 Unbounded islands

It is well known that every string graph has a representation by curves in which no three curves intersect in the same point, and in every intersection point the two curves sharing this point cross each other in this point (i.e., they do not touch). It can be easily seen that such a representation is homeomorphic to a system of curves that are piece-wise linear and their segments

follow the vertical and horizontal lines of a planar grid. Given such a representation of a graph G, one can rotate it by 45 degrees, and blow it up so that the segments of the representation follow the diagonal lines of the (now) extended grid in such a way that the segments intersect inside the grid squares but not in the grid points. Then one may represent each vertex of G by the island consisting of the grid points lying on the corresponding curve. If we further blow up so that the segments of the curves use only every second diagonal in each direction, any two islands are adjacent if and only if their corresponding curves cross. (See an illustration in Fig. 4.3.)

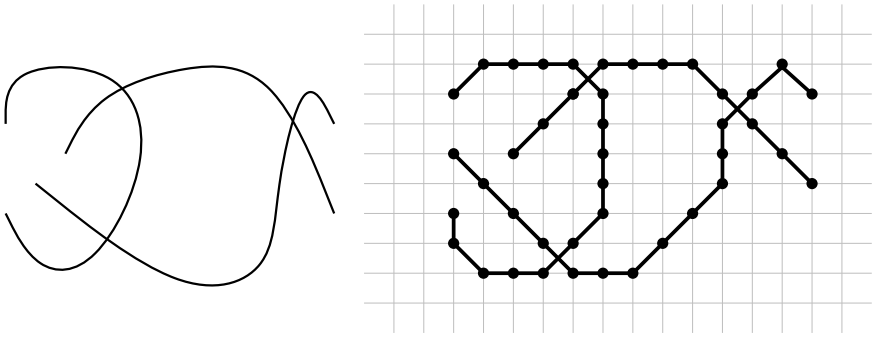


Figure 4.3: A string representation and an ISLAND representation of K_3.

On the other, it is also well known (cf. e.g. [43]) that string graphs are exactly the intersection graphs of arc-connected sets in the plane. Since for every island i the expanded set i^+ is arc-connected, it follows that every ISLAND graph is a string graph. Therefore we have the following claim.

Proposition 4.2.1. *The ISLAND graphs are exactly the string graphs. Hence the problem ISLAND is NP-complete.*

There is an even more straightforward way to transform a faithful representation of a graph in the extended grid into a string representation. For every island i (which represents a vertex $u_i \in V(G)$), consider a spanning tree T_i of the subgraph of the extended grid induced by the points of i. If two islands i, j are adjacent via a grid edge $e = pq$ (vertical, horizontal, or diagonal), with vertices $p \in T_i$ and $q \in T_j$, one can create a new vertex P_e in the middle (of the drawing of) e and add the edge pP_e to T_i and the edge qP_e to T_j. The newly added point P_e may belong to more than just these two trees (if e was a diagonal edge), but in such a case all the (at most) four

islands involved in the corners of the square whose diagonal is e are adjacent in the extended grid and so are the corresponding vertices in G. Hence G is the intersection graph of a family of trees in the plane. Replacing every tree by a curve running around the drawing of the tree at a very small distance we get a representation of G by intersecting curves, where every common point P_e will give rise to 2 crossing points of the curves, and every crossing of diagonal edges of two different trees gives rise to 4 crossing points. A k-island would thus result in a tree with a linear number of nodes (linear in k), and hence the total number of crossing points in the string representation obtained in this way would be $\mathcal{O}(nk)$. It is known that there are string graphs on n vertices requiring at least $\Omega(2^{cn})$ crossing points in any string representation [47]. Hence one obtains the following result.

Proposition 4.2.2. *There are ISLAND graphs which require exponentially many grid points in any faithful representation.*

We conclude that the natural guess-and-verify approach to obtain a faithful representation of a graph, by guessing the representing islands point by point, fails. This observation highlights the fact that the membership of ISLAND in NP is due to deeper reasons [62].

For the rest of the section we are interested in the bounded case, so to say about graphs representable by k-ISLAND for arbitrary but fixed k.

4.3 Orientability

In the following two sections we introduce problems which we use later as they are needed for our reduction from SAT.

Assume an instance ϕ of SAT. It contains set of clauses C and set of variables V. We create for such formula ϕ graph G^ϕ which has the following properties.

$$V(G^\phi) = \{c : c \in C\} \cup \{u^+ : u \in V\} \cup \{u^- : u \in V\},$$
$$E(G^\phi) = \{cu^+ : c \in C, u \in c\} \cup \{cu^- : c \in C, \bar{u} \in c\}.$$

The vertices of the graph represent clauses and literals. An edge represents that the literal is a part of the clause. We say that G^ϕ is orientable if there is an orientation of every edge such that outdegree of clause vertices is at least 1 and for each pair of literals u^+ and u^- indegree of u^+ or u^- is 0.

Breu and Kirkpatrick in [17] proved the following lemma, which is a simple property of orientability.

Lemma 4.3.1. *A formula ϕ is satisfiable if and only if G^ϕ is orientable.*

4.4 Graph Embedding on the Grid

Let G^ϕ be a graph from the previous section. We use the construction of Breu and Kirkpatrick [17], and here we describe properties of such embedding.

We embed G^ϕ into a grid of size $(6|V|+1) \times (3|C|+2)$. Every vertex of the grid is either unused or associated with component of the embedding and bounded by centered (in the vertex) unit square. There are 3 groups of components, literals, clauses and communication components. There are 3 groups of communication literals: wires, corners, cross-overs. The *wire* is a component containing a unit length line segment which overlaps the grid vertex. The *corner* is two half unit segments which meet in right angle at a grid vertex and the *cross-over* is two unit length segments which cross in the right angle in the grid vertex. There are 2 kinds of wire components (vertical/horizontal) and 4 kinds of corner components but just one cross-over component.

Each component contains one to four terminals. Terminals terminate line segments. Terminal T is on the top side of component's unit square. Terminals B, L and R on the bottom, left and right sides of the component's unit square, correspondingly. Component a is adjacent to component b if one of their terminals overlap. Terminals are represented by small black circles in Fig. 4.4. We have one exception, which is a complementary pair of literal components, which is supposed to have a terminal in between but we do not show a terminal between them as they form a single truth setting component.

The orientation of a terminal is direction. We call terminal T (resp. B, L, R) *directed away* if it is oriented N (resp. S, E, W), *directed towards* otherwise. We call the graph orientable if every terminal satisfies following rules:

1. Only terminals adjacent to a vertical line are directed N or S.
2. Only terminals adjacent to a horizontal line are directed E or W.
3. Every wire and corner and clause have at least one terminal directed away.

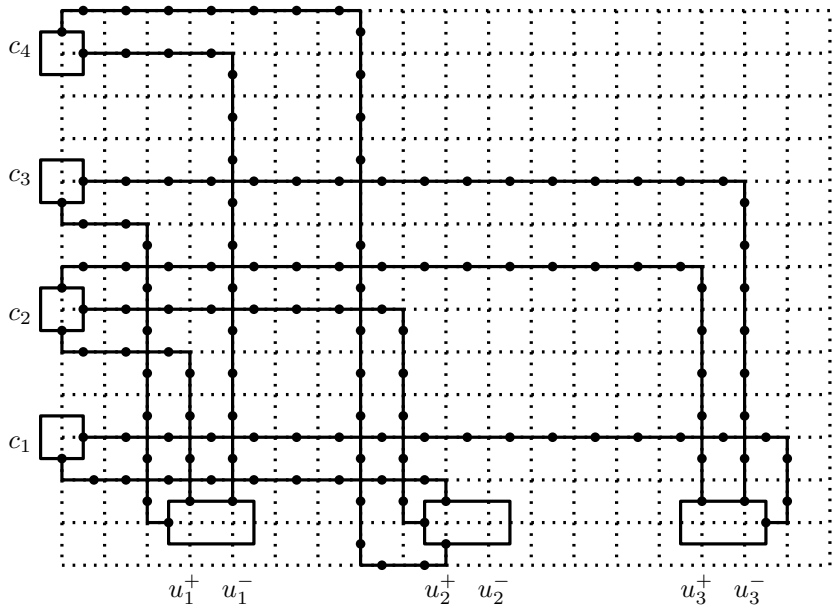


Figure 4.4: A grid embedding of the incidence graph. The graph corresponding to **SAT** instance $U = \{u_1, u_2, u_3\}$ and $C = \{\{u_2, \overline{u_4}\}, \{u_1, u_2, u_3\}, \{u_1, \overline{u_3}\}, \{\overline{u_1}, u_2\}\}$

4. Every cross-over line segment contains at least one terminal directed away.

5. Every truth setting component has one literal such that all terminals are directed away.

With this components one can produce a drawing G_D^ϕ of graph G^ϕ which is shown in Fig. 4.4. Breu and Kirkpatrick [17] proved the following using the described model.

Lemma 4.4.1. *A grid drawing is orientable if and only if the underlying graph is orientable.*

If we put together lemmas 4.4.1 and 4.3.1 we get the following.

Corollary 4.4.2. *A grid drawing is orientable if and only if the underlying instance of* SAT *is satisfiable.*

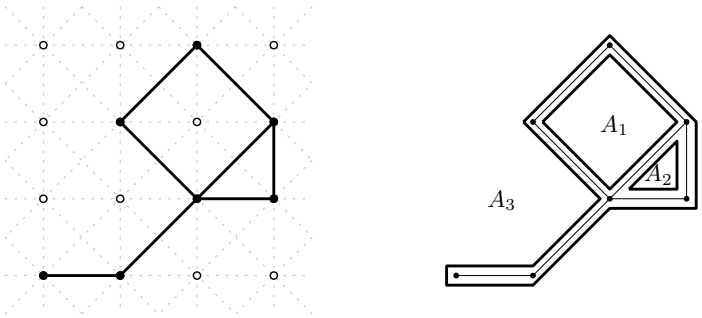


Figure 4.5: On the right side, the set of black vertices is a configuration S and black edges represent its induced subgraph. On the right side there is cutting of the Δ-graph defined by S, where A_1, A_2 and A_3 are regions of the cutting.

4.5 The Basic Building Block

We often use together graph and its representation, therefore we introduce the following notation. Δ-graph is a tuple (G, D) where G is a graph and D is its embedding. For example, in this chapter we always assume that the underlying grid is Δ-graph, where the vertices have integral coordinates and edges are straight lines. By a configuration of vertices we mean a subset S of vertices of the grid.

Definition 4.5.1. *Let (G, D) be Δ-graph. We define the cutting as the set of regions created by the partition of plane alongside the drawing (of edges of) D of G. Let S be a configuration and G be a Δ-graph induced by S in the grid, then A is a region of S if A is in the cutting of G, see Fig. 4.5.*

Definition 4.5.2. *The capacity of region A is the number of the inner vertices (points with integral coordinates) of A (we do not count vertices on the boundary of A), which is denoted by $c(A)$. A maximal region is a finite region with the maximal capacity (we exlude the outer region). The capacity of S is the capacity of a maximal region, which is denoted by $c(S)$. Configuration S is maximal if*

$$c(S) = \max\{c(Z) : Z \text{ is a configuration and } |Z| = |S|\}.$$

We are now going to explore properties of configurations.

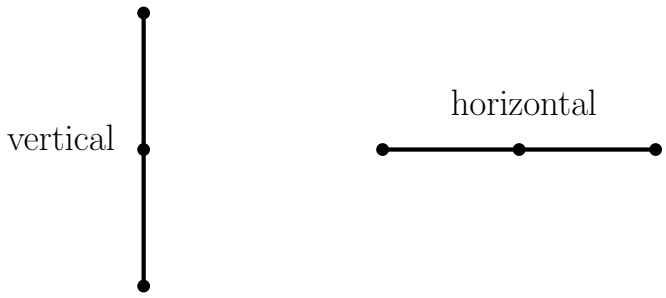


Figure 4.6: The straight sub-configuration

Lemma 4.5.1. *Every maximal region of maximal configuration S is convex. (Set C is convex if $x, y \in C$ then every point $xk + y(1-k) \in C$ for $k \in [0,1]$).*

We just sketch a proof. Assume that there is a maximal region A which is non-convex, then we show how to define a region with a bigger capacity. If A is non-convex, then there exist 3 consecutive vertices on the boundary of A which already witness non-convexity. There are only two possible cases. We increase capacity either by removing one vertex from these 3 or we flip these vertices "out". At least in one of these cases we get a bigger capacity of a new region. We leave the details to a careful reader.

Essentially the same proof can be applied to the following lemma.

Lemma 4.5.2. *Let S be a maximal configuration and A be a maximal region, then A does not contain a consecutive triple of vertices on the boundary from Fig. 4.6.*

The following simple lemma shows a monotonic property of capacities.

Lemma 4.5.3. *Let S and Z be maximal configurations, where $|S| > |Z|$ then $c(S) \geq c(Z)$.*

Proof. Let maximal region of Z be A then create configuration S' such that S' contains all vertices of Z and we add an extra set V with $|S| - |Z|$ vertices into S' in such a way that $V \cap A$ is empty and S' induce a connected graph, that is clearly always possible. That already concludes the claim. □

Lemma 4.5.4. *Let A be a maximal region of maximal configuration S, then boundary of A induces a connected graph.*

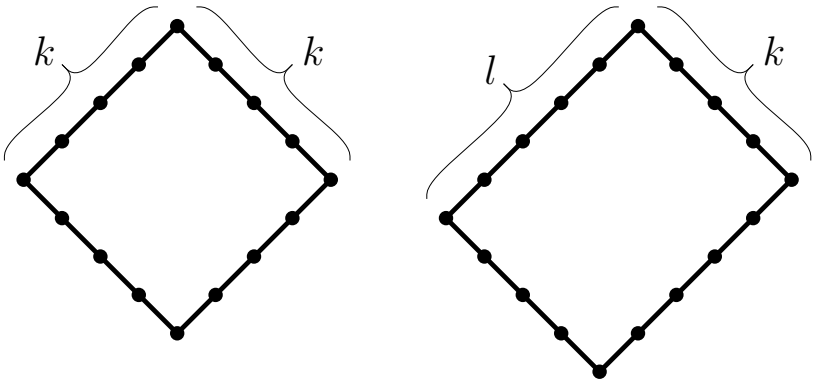


Figure 4.7: A diamond Δ-graph and a rectangular Δ-graph.

Proof. Assume for contradiction the boundary does not induce connected graph. Then the boundary of A is formed by k connected components $C_1, C_2, \ldots, C_k$. Since A is a finite region, there is i such that C_i divides plane into inner part and outer part and all other components are in the inner part. Assume $l \leq k$ and $k \neq i$, then if we remove vertices of C_l from S, we get a bigger capacity of the maximal region and together with Lemma 4.5.3 it gets us to contradiction. □

Definition 4.5.3. *A rectangular Δ-graph with the side-lengths k and l vertices is graph G and its drawing D where G is a cycle on $2k+2l$ vertices and for D see Fig. 4.7. If k and l are same we call it a diamond Δ-graph.*

Later we will use the following observation, which is not stated formally.

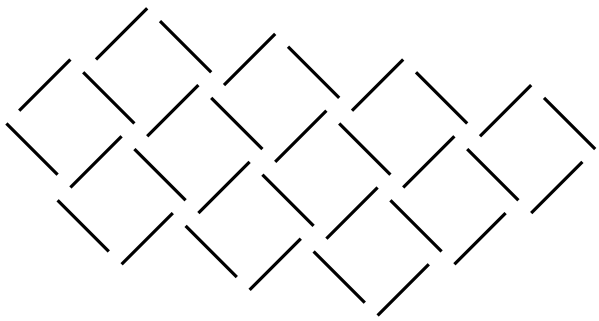


Figure 4.8: The diamond tiling Δ-graph.

Observation 4.5.5. *It is possible to cover the whole plane with copies of the diamond graph forming pattern depicted in Fig.4.8.*

Here comes the most complicated and also the most important lemma.

Lemma 4.5.6. *Let S be a maximal configuration on $4n$ vertices, where $n \geq 1$. Then S induces a diamond Δ-graph.*

Proof. Let A be a maximal region of S. From Lemma 4.5.4 we know the boundary of A induces a connected graph. We explore the boundary of R assuming an ordering $o = (e_1, e_2, \dots e_k)$ of edges on boundary of A, where the ordering is going clock-wise around A. Let e_i be an edge on the boundary of A. We want to know which kind of an edge can be consecutive e_{i+1} edge. For this purpose the case analysis follows.

To simplify notation we use small pictures with an oriented edge and a dot, the orientation in picture defines the direction of o, and the slope of edge defines the type of edge and a dot defines where the inner part of A is. For example,  defines a diagonal edge and the direction of ordering o and region A is on its right side. Assume first edge e_i to be . Our question now is which kind of edge e_{i+1} can follow edge e_i in o? We analyze all possible cases and exclude some, see Fig. 4.9. Cases (a)-(c) do not form a maximal region of a maximal configuration. Since the cases are not convex, that contradicts Lemma 4.5.1. Case (g) forms the region of zero capacity (the dashed edge is induced) and we can notice that there is a configuration of size 4 with capacity 1. It means that the region of case (g) is not a maximal region of any maximal configuration.

Now assume edge e_i to be  . We again explore what possible embeddings of edge e_{i+1} are, see Fig. 4.10. Cases (a)-(c) contradict Lemma 4.5.1, case (d) contradicts Lemma 4.5.2, and cases (f) and (g) contradict a maximality of R and S, therefore the only possible case is (e).

All other cases of edges e_i are just rotations (by 90°, 180° and 270°) of two previous types. By case analysis it can be shown (we avoid it here), that the boundary of R induces a rectangular Δ-graph or a rectangular Δ-graph with some corners cut off, see Fig. 4.11.

Assume that a maximal region has $4k$ vertices on the boundary. Then we can easily count that the capacity is strictly maximized if the boundary induces a diamond Δ-graph. Alternatively we can say that if A is a maximal region of S and the number of vertices on the boundary is a multiple of 4

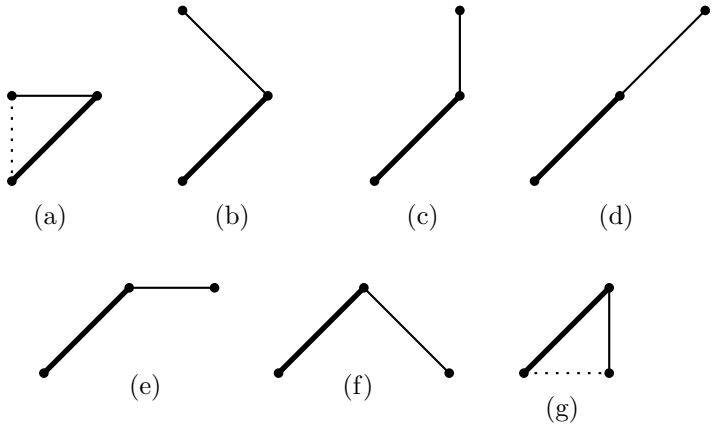


Figure 4.9: All cases, where e_i is the following thick edge.

and the boundary does not induce a diamond Δ-graph then configuration S is not maximal.

Assume now for a contradiction that S is a maximal configuration with $4n$ vertices which does not induce a diamond Δ-graph. Then consider a maximal region A of S. If A induces a diamond Δ-graph then either S is equal to vertices on the boundary of A or the size of the boundary of A is smaller than $|S|$ and then S is not maximal because there exists the configuration which induces a diamond Δ-graph which has a bigger capacity. Therefore, if the boundary of A does not induce the diamond Δ-graph then it contradicts what we proved above. □

A straightforward consequence of the previous proof, the monotonic property of capacities, and a little bit of counting (which we omit here) yield the

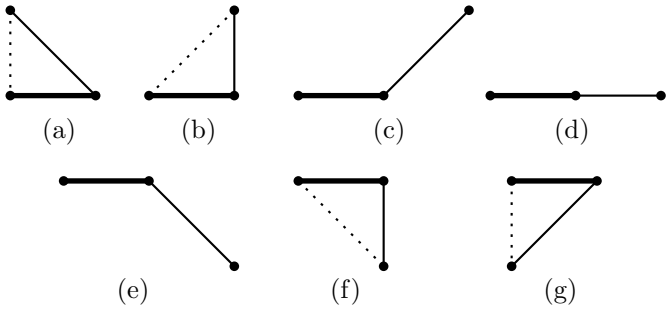


Figure 4.10: All cases, where e_i is the following thick edge.

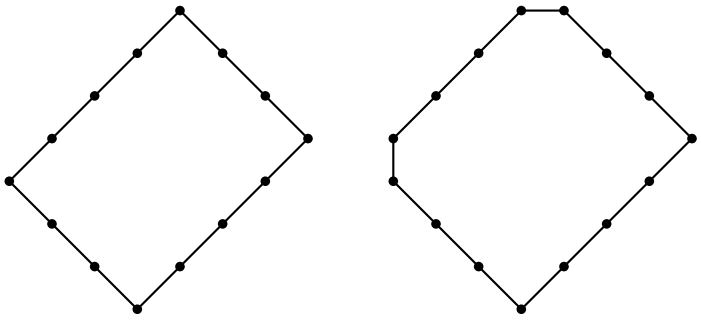


Figure 4.11: Examples of a maximal region.

following corollary.

Corollary 4.5.7. *Let R be a maximal region of a maximal configuration S, where $S = n$. Then $c(R) \geq 2\lfloor \frac{n}{4} \rfloor^2 - 2\lfloor \frac{n}{4} \rfloor + 1$.*

4.6 Basics of Island Representations

Let R be an island representation of G and B be a subgraph of G then $R|_B$ is the island representation of B within R. Because any island representation Q defines a configuration S (union of all islands of Q), we use notion of capacity, cutting, etc. also for island representations. In the previous section we showed that a configuration of n points cannot "wrap" more than some certain amount of points. We also find out how maximal configurations look like. In this section we explore island representations based on our understanding of configurations.

For the rest of the chapter we assume k to be arbitrary but fixed and we refer to a k-island simply as an island.

Cage. Let D be the diamond Δ-graph of side length $k \cdot c$, where c is an appropriate constant, whose choice will be clear later. Let S be a configuration inducing D. We partition vertices of S into islands of size k, such that k consecutive vertices start from an arbitrary corner, for details see Fig. 4.12. We refer to this set of islands as cage of side length c.

Flipper. Let D and S be the same as before. Assume cutting C of S and R be the inner region of the cutting C. All integral vertices of R (in this case we assume also boundary vertices) define islands and we refer to this set of islands as flipper of side length c. And again for details see Fig. 4.12.

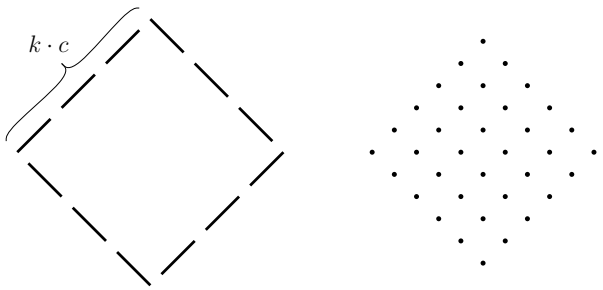


Figure 4.12: An island representation of the diamond graph and flipper graph, when $c = 3$

The idea behind these gadgets is simple, we need to "wrap" a flipper into the cage in such a way that we can "wrap" one but not more.

Lemma 4.6.1. *Let H be a graph and G_1 and G_2 be subgraphs of H, where G_2 is connected and G_1 is disjoint and non-adjacent to G_2. Let R be an arbitrary island representation of H then $R|_{G_2}$ is contained in a single region of cutting by $R|_{G_1}$.*

Proof. Assume there are two regions A_1 and A_2 from cutting of $R|_{G_1}$ which contain some vertices of $R|_{G_2}$. Assume that v_1 and v_2 are vertices of $R|_B$ and $v_1 \in A_1$ and $v_2 \in A_2$. Define Q as Δ-graph induced by $R|_{G_2}$. Then there is a path p in Q from v_1 to v_2 which crosses the boundary of A_1, which already leads to contradiction because G_2 is disjoint and non-adjacent to G_1. □

Definition 4.6.1. *Let C_n be a cycle, R be an island representation of C_n and H be Δ-graph induced by R (we use R also as a configuration). A simple closed curve is called a cycle curve if it is a subset of a drawing of H and contains at least one vertex from each island.*

Lemma 4.6.2. *For each island representation R of C_n (cycle with n vertices), where $n \geq 3$, there is a cycle curve $\mathcal{C}$.*

Proof. Assume ordering of vertices of C_n as they appear on the cycle. The i-th vertex is denoted by v_i and the island corresponding to v_i in R is V_i.

We use here an iterative definition for vertices a_i. We start by choosing an arbitrary vertex a_1 from the representation of V_1 adjacent to V_2, as there always exists one. Assume that we already defined vertices $a_1, \ldots a_i$. Let H_i be a Δ-graph induced by V_i. Using graph distance metric, we find

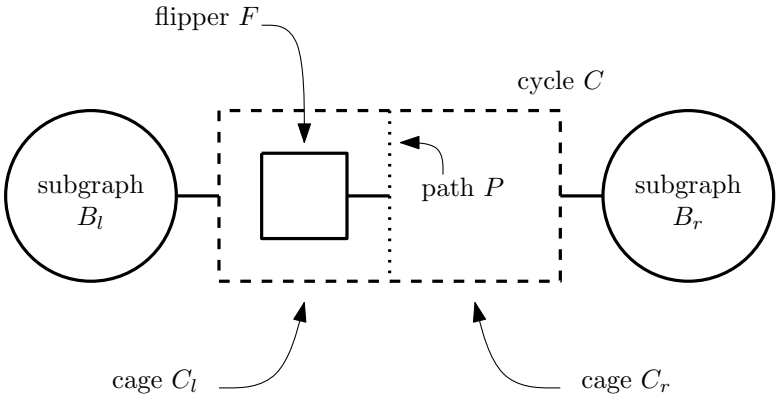


Figure 4.13: the switch graph W

vertex $a_{i+1} \in V_{i+1}$ closest to vertex a_i within the graph induced on vertices $V_i \cup \{a_{i+1}\}$. Let path p_i be the shortest path between a_i and a_{i+1} found as described above. Repeating this procedure until we define vertex a_n yields shortest path p_n within H_n that connects a_n and a_1. Assume that $u \in p_n$ is the first vertex adjacent to p_1, in that case we redefine path p_1 as a subpath of p_1 from u to a_2 and path p_n as subpath of p_n from a_n to u. Paths $p_1, p_2, \ldots p_n$ already define a cycle curve $\mathcal{C}$.

We choose in each step of the algorithm the shortest path, therefore the path created as $p_1, \ldots, p_{n-1}$ defines a simple curve. Path p_n cannot intersect p_{n-1}, otherwise we would choose vertex a_n differently. And p_n does not intersect p_1 because we redefined it that way. □

That suggests us the following definition.

Definition 4.6.2. *Let R_G be an island representation of graph G and let ϕ be a cycle curve of cycle C. If R_G is completely contained in the inner region defined by ϕ, then we say that R_G is contained in ϕ. If it is clear which representation of G and C we refer to, we can even say that G is contained in C.*

If two graphs A and B are connected by path P_2 (of length 2) we say that A and B are connected by a *hinge* and hinge refers to path P_2.

Next, we are interested in so called switch graph W. W is schematically shown in Fig.4.13. It contains two cages C_l and C_r which share path P and the outer cycle defines cycle C. Flipper F is hinged to P and there are two "big" enough connected graphs B_l and B_r connected by hinges to cycle C.

By big enough (for graphs B_l and B_r) we mean the graphs which in any representation cannot be contained in any representations of C_l or C_r.

Lemma 4.6.3. *Let ϕ be a cycle curve of C in an island representation R of the switch graph W. Let I be the inner region and O be the outer region defined by ϕ. Then the large enough connected subgraphs B_l and B_r are represented by islands within the outer region O.*

Proof. Assume it is not the case. Let H be the Δ-graph induced by configuration R. Then ϕ defines a subgraph of H. From Corollary 4.5.7 and from Lemma 4.6.1 we get that the connected subgraphs B_l and B_r cannot be embedded in the region I. □

The next lemma shows even more properties of an island representation of switch graph W.

Lemma 4.6.4. *Let's W, R, C, I, O, ϕ and P be defined as in Lemma 4.6.3 and the definition of the switch graph W. Then $R|_F$ is contained in the bounded plane region I.*

Proof. Assume that it is not the case, that is, $R|_F$ is not contained in I. As F is connected and ϕ defines a cycle curve then Lemma 4.6.1 forces $R|_F$ to be either inside or outside of ϕ. Therefore, $R|_F$ is inside of region O defined by ϕ. Thus, path P (except for vertices adjacent to C) has to be represented inside of region O. There are several analogical cases depending on representation of path P and location of flipper F. We provide here just one, see Fig 4.14, as the other are very similar. Let H be the subgraph defined by $C \cup P$ of W. Let's assume that cycles C_l and C_r are defined as in the definition of the switch graph. Lemma 4.6.1 implies that either B_l or B_r is contained in either C_l or C_r, which contradicts Corollary 4.5.7 and that already leads to contradiction. □

4.7 Gadgets

We already introduced all necessary tools, so now it is the time to show a reduction from the orientability. The first step is to define gadgets which represent an orientable graph. Luckily, all our gadgets are based on the switch graphs and that makes it simpler. We do not define gadgets through

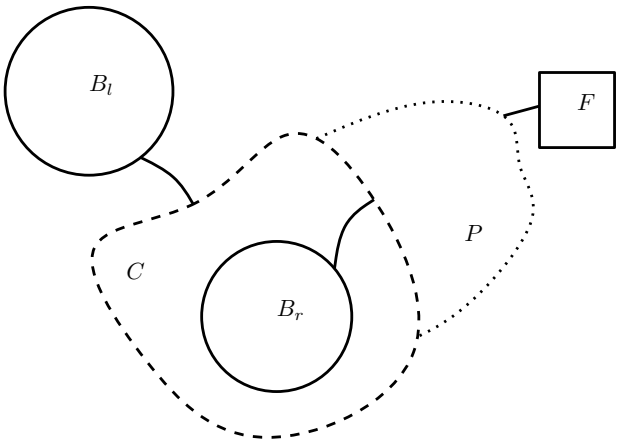


Figure 4.14: a possible representation of W

the graphs but through their island representation. The advantage is that one can immediately see that these graphs have feasible representations.

As we observed in Observation 4.5.5 we can tile the whole plane with diamond graphs, see Fig. 4.8. This particular tiling defines our coordinate system which we use when constructing our gadgets. We construct gadgets in such a way that they fit (with an exception for flippers which have "freedom" of representation within the coordinate system) into 5×5 block of the plane.

Link gadget. The simplest gadget is called the *link* gadget. It is composed out of 5 cages and 5 flippers, as we promised. We omit a formal definition and rather refer to Fig. 4.15. The dashed square defines 5×5 region.

An important property of the link gadget is that in any island representation R, exactly one flipper is represented inside and one is represented outside.

One can notice that we assign a number to every flipper in the representation, that represents an approximate volume of the flipper according to a cage. It is approximate because exact calculation would not be very illustrative. We use this convention for all other gadgets as well.

Literal gadget. Next we introduce the *literal* gadget, see Fig. 4.16. Let I be an arbitrary island representation. Then if the right-most flipper (right-most in Fig.4.16) is represented inside then the left-most, up-most, down-most flippers are outside. And similarly, if any of up-most, left-most or down-most are represented inside then right-most is represented outside.

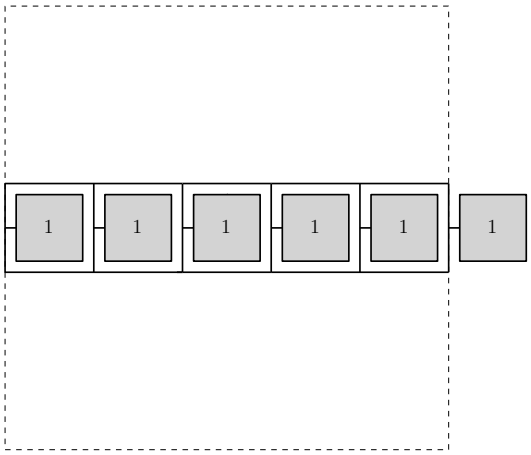


Figure 4.15: The link gadget.

Variable gadget. Now we define the *variable* gadget which is made out of two literal gadgets, see Fig. 4.17. The properties of this gadget are a straightforward consequence of properties of the literal gadget.

Clause gadget. The *clause* gadget is shown in Fig. 4.18. Its properties are what one would expect: at least one flipper has to be out in any representation.

Cross-over gadget. By far the most complicated gadget is the *cross-over* gadget, see Fig. 4.19. This gadget emulates two link gadgets which cross but stay independent. More precisely, if the left-most flipper is inside, then the right-most flipper has to be outside, but the up-most flipper can be inside or outside and according to its position, the position of the down-most flipper can be determined.

There is the last gadget which we are going to use, but this one is very simple, we call it the *corner* gadget and has same properties as the link gadget except it changes direction.

4.8 Construction

In Section 4.4 we defined graph G_D^ϕ. Graph G_D^ϕ is composed of vertices associated with 3 groups of components: literal, clause and communication components. We now transform graph G into new graph H, which has k-island representation if and only if G_D^ϕ (and also G^ϕ) is orientable.

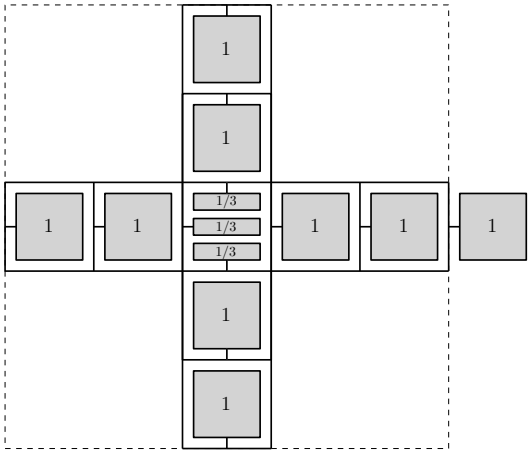


Figure 4.16: The literal gadget.

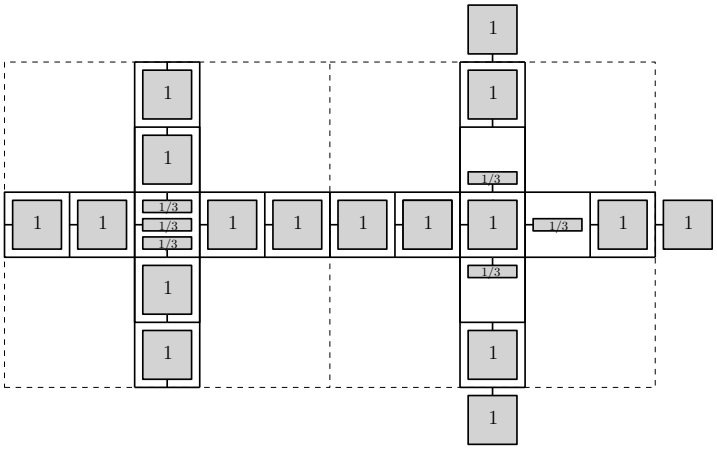


Figure 4.17: The variable gadget.

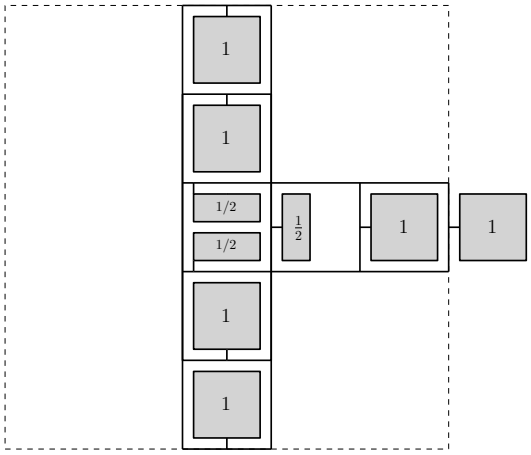


Figure 4.18: The clause gadget.

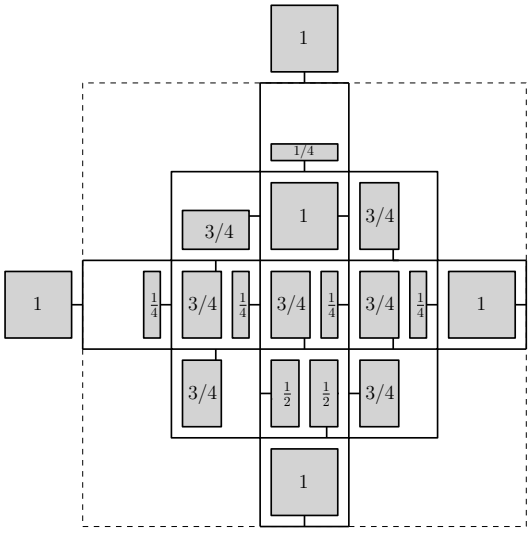


Figure 4.19: The cross-over gadget.

Every gadget has size 5×5 (if we do not have flippers and do not count a variable gadget as two gadgets). To construct graph H, for every vertex associated with c component in G_D^ϕ we put into H gadget c, where c can be variable, clause or communication (link, corner, cross-over). We add gadgets to H in such a way that adjacent gadgets share one side of a cage and a flipper.

Lemma 4.8.1. *Let graphs H and G be defined as above. Then G is orientable if and only if H has an island representation.*

Proof. The first implication is simple: if G is orientable then H has an island representation. According to the orientation, we set flippers in the gadgets of H and from the definition of H we can use representations from definitions of our gadgets. That already concludes the first implication.

Assume that we have an island representation of H. We have to define how to encode orientations of edges of G from gadgets of H.

Orientations of graph G are based on the drawing of flippers on the boundary of the gadgets. Each gadget has up to 4 cages whose one side is shared with another gadget. The side of cages is connected to a flipper and we call this flipper the boundary flipper. We define the orientation according to the position of boundary flippers. If flipper f is a boundary flipper for gadgets representing vertices u and v and is contained in a cage associated with the gadget representing vertex v then we set the orientation from u to v and opposite otherwise. From properties of gadgets proven above this already defines orientation. □

Chapter 5

Modifications of Graphs

There are always disproportions between theory and applications. In theory one can assume nice properties about your object and with that assumption one can solve the problem. In applications it often happens that some assumptions are not met. so we cannot use directly the theoretical results obtained under those assumptions.

This is already very close to what graph modifications are good for. We assume that our object is a graph and we would like to find the smallest possible modification to the graph so that it has given properties. A *graph property* is a set Π of graphs. Any graph G in Π is a Π-graph, or we also say that G has property Π. We would like to find how many edge deletions, vertex deletions or edge contractions we need to modify an input graph to become Π-graph.

In particular, our strongest interest lies in properties related to planarity.

5.1 Vertex and Edge Deletion

Already in the classic book of Garey and Johnson [28] on NP-completeness, the authors formulate 18 graph modification problems. And the number of interesting graph modification problems is growing. Notice that many well-known problems can be reformulated in such a way that hey become graph modification problems. For example graph, connectivity can be seen as removing the minimal number of vertices so that the graph becomes disconnected. Similarly, the maximum clique problem can be seen as the problem of finding the minimal number of vertices so that the graph becomes a complete

graph.

In the rest of the section, we are exploring the following two problems:

Problem:	Π-Vertex Deletion
Input:	An undirected graph $G = (V, E)$ and integer k
Output:	Is there a set of vertices $V' \subset V$ of size at most k such that the graph induced on vertices $V \setminus V'$ is Π-graph?

Problem:	Π-Deletion
Input:	An undirected graph $G = (V, E)$ and integer k
Output:	Is there a set of edges $E' \subset E$ of size at most k such that the graph $G' = (V, E \setminus E')$ is Π-graph?

There was a significant effort in proving general complexity theorem covering many different properties of graphs rather than proving them just for one property at a time. Therefore, there were introduced certain types of properties which have to be satisfied. A graph property is *non-trivial* if there is at least one graph which has this property but not all graphs have it. A graph property is *interesting* if there is an infinite number of graphs with the property and there is an infinite number of graphs without the property. The property Π is hereditary if it is preserved under vertex deletion.

In 70's Lewis and Yannakakis [50] proved the following general theorem:

Theorem 5.1.1. *For every non-trivial interesting hereditary property* Π*, the* Π-Vertex Deletion *problem is NP-hard.*

Due to this theorem, the complexity of Π-Vertex Deletion for all hereditary properties was set. Unfortunately, many simple properties are not hereditary. For example, an induced subgraph of tree is not necessarily a tree but a forest. Later the extension of Theorem 5.1.1 was proved by Yannakakis [71] to solve the noted issues as follows.

Theorem 5.1.2. *For every non-trivial interesting property* Π *that holds on every connected induced subgraph,* Π-Vertex Deletion *problem is NP-hard.*

You might notice that if the property Π is effectively recognizable then Theorems 5.1.1 and 5.1.2 imply NP-completeness of given problems.

There are no general results known for Π-Deletion problem. The reason might be because many problems of Π-Deletion are computationally simpler than Π-Vertex Deletion problem. This can be illustrated by property Π being a forest, where Π-Vertex Deletion is NP-hard from Theorem 5.1.1 but on other hand, the edge deletion version is trivially polynomial. Even though there are difficulties which we just noted, there were established some general results:

Theorem 5.1.3. *If a graph property* Π *is hereditary on subgraphs and is determined by 3-connected components then* Π-Deletion *problem is NP-hard.*

We define that property Π is *determined by 3-connected components* as follows. If a graph has property Π then also all its 3-connected components have this property. Another paper that provides a very general result about the minimum Π-Deletion is due to Alon et al. [2]:

Theorem 5.1.4. *Let* Π *be a graph property such that it holds for every bipartite graph then* Π-Deletion *problem is NP-hard.*

For our particular interest in planarity, notice that if property Π is defined as being planar then if we assume Π-Deletion problem then it is NP-hard from Theorem 5.1.1 and if we assume Π-Vertex Deletion then it is also NP-hard. This was independently shown by [70, 67, 31].

5.2 Edge Contractions

Another graph modification involves edge contractions and this section is devoted to those.

Comparing to other graph modification problems, it does not receive that much attention. In the very first papers Asano and Hirata [6] and Watanabe, Ae and Nakamura [67] showed a general result which implies NP-hardness for several classes of graphs. Of special interest for us is that Π-Contraction is NP-hard for Pi being a class of planar graphs, which was proved in [6].

Another direction of deals with the problem of contracting an input graph to fixed target graph H, which is called H-Contractibility. A reader can notice that if G is a part of the input then the problem is even harder

than the graph isomorphism problem. Brouwer and Veldman [18] proved the following:

Theorem 5.2.1. *H-Contractibility is NP-complete if H is a connected triangle-free graph other than a star.*

In [18] it is also proved that for any connected graph with at most 4 vertices H-Contractibility is polynomial with exception for P_4 and C_4 for which H-Contractibility is also NP-complete. There came an extension of [18] for small graphs by Levin et al. [49]:

Theorem 5.2.2. *Let H be a undirected graph on at most 5 vertices. If H contains a dominating vertex then H-Contractibility is solvable in polynomial time, otherwise H-Contractibility is NP-complete.*

There are also several results when H is assumed to be a graph from a well-known graph class. Belmonte et al. [11] showed that there is a polynomial time algorithm for H-Contractibilityfor H being a chordal graph. Golovach et al. [32] provided the same result for the class of split graphs. Finally, Kamiński et al. [41] extended the results for the class of planar graphs, which is of our strongest interest.

For the last several years, the research on Π-Contraction from perspective of parametrized complexity has brought more attention. We keep the same notation for problems, but in parametrized complexity setting we mean the following:

Problem:	Π-Contraction
Input:	An undirected graph $G = (V, E)$ and integer k
Output:	Is there a set of k edges such that contracting them makes a graph a Π-graph?
Parameter:	k

One of the first FPT algorithms was due to Hegges et al. [38], who provided an FPT algorithm where Π is the class of bipartite graphs. This algorithm runs in time $2^{2^{\mathcal{O}(k^2)}}n^{\mathcal{O}(1)}$. Their approach uses the theory of graph minors which was developed by Robertson and Seymour in a series of papers. In particular, it uses the *irrelevant vertex* method introduced in [60]. Later, the result for bipartite graphs was improved by Guillemot and Marx [36], who developed conceptually simpler algorithms.

Theorem 5.2.3. Π-Contraction *for* Π *being the class of bipartite graphs has a randomized* FPT *algorithm with running time* $2^{\mathcal{O}(k^2)}nm$ *and a deterministic algorithm with running time* $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$.

Hegges et al. [39] showed an FPT algorithm running in $4.98^k n^{\mathcal{O}(1)}$ time if Π is the class of all trees. If Π is the class of paths then they showed an FPT running in $2^{k+o(k)} + n^{\mathcal{O}(1)}$. Golovach et al. [33] were studying the case when Π are graphs of minimum degree d. They proved that this problem is fixed-parameter tractable when both k and d are parameters. In particular, their algorithm has running time $2^{\mathcal{O}(k \log d)}n^{\mathcal{O}(1)}$. When either d or k is a part of the input then the problem becomes hard in the following sense: If we parameterize only by d or only by k, then the problem becomes hard. They proved that the problem is NP-complete even if $d = 14$ and W[1]-hard when only k is the parameter. The most recent Π-Contraction problems for Π being planar graphs were shown to have a polynomial time FPT algorithm. It was provided independently by Golovach et al. [34] and by our group. We present our result in the next section.

5.3 Our Result

We study in this section Π-Contraction, where Π is a planar graph. Notice if the number of contracted edges is not limited, every connected graph can be trivially contracted into a single vertex, and thus becomes planar. A graph is *k-contractible* if the number of contracted edges is limited by a number k. If k is a part of the input, testing k-contractibility is NP-complete due to Asano and Hirata [6]. We present here a fixed-parameter tractable algorithm for contractability to planar graphs.

Definitions and Notation. We denote by $G \circ e$ the graph obtained by

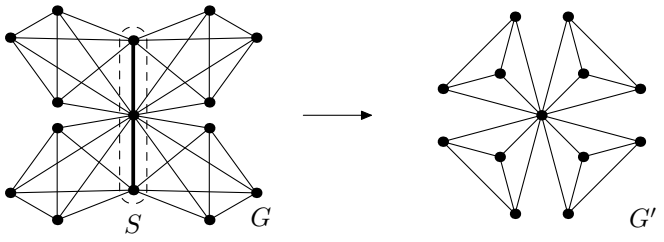


Figure 5.1: An example of a 2-contractible graph.

contracting an edge e in G. For a set of edges S, we denote a graph created from G by contracting all edges of S by $G \circ S$. We call $S \subseteq E$ *a planarizing set* of G if $G \circ S$ is a planar graph. We say that G is *k-contractible* if there exists a planarizing set S of size at most k.

MSOL formulas for graphs are logic formulas which contain predicates of equality, incidence and containment, logic operators and quantifiers for vertices, edges, sets of vertices and edges. For instance, 3-colorability can be expressed by MSOL as existence of three sets V_1, V_2 and V_3 of vertices, such that each vertex belongs to exactly one V_i, and there are no edges with both endpoints in one V_i.

Restricted Contractibility. We address the following more general problem. A subset of edges S is called a *planarizing set* if by contracting S in G the graph becomes planar; see Fig. 5.1. We want to find a planarizing set S of size at most k that satisfies an additional restriction: a monadic second-order logic (MSOL) formula $P(S, G)$ fixed for the problem.[1]

Problem:	P-RestrictedContract
Input:	An undirected graph G and an integer k.
Output:	Is there a planarizing set $S \subseteq E(G)$ of size at most k satisfying $P(S, G)$ that when contracted produces a planar graph?

We want to construct an FPT algorithm for this problem with respect to the parameter k. This is not possible for every MSOL formula $P(S, G)$. For some formulas, the problem is already NP-hard even for $k = 0$. For instance, let $P(S, G)$ be the formula: Is there the empty set S such that after contracting S the graph G is 3-colorable? Then the problem P-RestrictedContract is equivalent to testing 3-colorability of planar graphs which is known to be NP-complete [29].

In this section, we describe an FPT algorithm which works as follows. Either the graph is simple (of small tree-width) and the problem can be solved in a brute-force way. Or we find a small part of the graph which we can prove to be far from any inclusion-minimal planarizing set. We modify the graph by contracting this small part, and repeat the process. Therefore,

[1] More precisely, we have different formulas $P_k(e_1, \ldots, e_k)$ for each k where $S = \{e_1, \ldots, e_k\}$. So the length of the formula may depend on k.

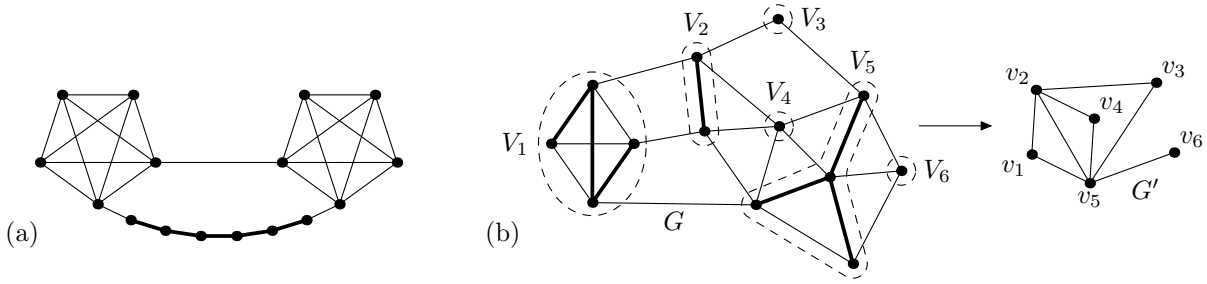


Figure 5.2: (a) Every inclusion-minimal planarizing set S contains one edge in each K_5. Therefore, inert sets are subsets of the highlighted edges. (b) The connected components of a planarizing sets S act as clusers. After contracting S, each cluster corresponds to one vertex in G'. Two vertices in G' are adjacent if and only if there exists an edge in G between the corresponding clusters.

we need to restrict ourselves to MSOL formulas for which satisfiablity is not changed by this modification.

An MSOL formula P is *inclusion-closed* if for every S satisfying P also every $S' \subseteq S$ satisfies P. This property is necessary since the algorithm looks for inclusion-minimal planarizing sets. A set of edges B of G is called *inert* if it is not incident with any edge of any inclusion-minimal planarizing set S; see Fig. 5.2a for an example. A formula P is called *inert contraction-closed* if following holds for every inclusion-minimal planarizing set S and every inert set B

$$P(S, G) \iff P(S, G \circ B)$$

Therefore the modification by contraction is not changing solvability of the problem.

Theorem 5.3.1. *For every inclusion-closed and inert contraction-closed MSOL formula P, the problem P-*RestrictedContract *is solvable in time $\mathcal{O}(n^2 f(k))$ where n is the number of vertices of G.*

Our algorithm uses an approach developed by Grohe [35] which shows that there is quadratic-time FPT algorithm for finding the crossing number. The most significant difference is the proof of Lemma 5.6.1. We cannot use the same approach as that of [35] because k-contractible graphs do not have bounded genus, which you can find proved in [34], which is essential in [35].

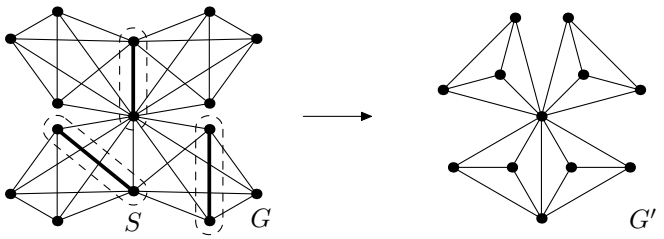


Figure 5.3: For $\ell = 2$, when we require that a planarizing set S is a matching, a smallest planarizing set contains three edges.

Further, our approach from the proof of Lemma 5.6.1 can be modified for the crossing number which simplifies the argument of Grohe [35]; see Section 5.8.

For a trivial formula P that is true for every set of edges, we get the k-contractibility problem considered above. We note that k-contractibility was independently proved to be solvable in time $\mathcal{O}(n^{2+\varepsilon}\bar{f}(k))$ for every $\varepsilon > 0$ in a recent paper of Golovach et al. [34]. The algorithm described here uses similar techniques but improves the time complexity and in fact solves a more general problem.

ℓ-subgraph Contractibility. For different formulas P, we get problems different from k-contractibility having new specific properties. As one particular example, we work with a problem which we call ℓ-subgraph contractibility. A graph is called *ℓ-subgraph contractible* if there exists a planarizing set S such that its edges form disjoint connected subgraphs with at most ℓ vertices. For instance, for $\ell = 2$ the planarizing set S is required to be a matching, see Fig. 5.3.

Problem:	ℓ-SubContract
Input:	An undirected graph G and an integer k.
Output:	Is G ℓ-subgraph contractible by a set S having at most k edges?

Contraction of a set S can be interpreted as graph clustering, see Fig. 5.2b. We want to find clusters such that the resulting cluster graph is planar. For ℓ-subgraph contractability, every cluster has to be of size at most ℓ. In comparison to k-contractibility, the contracted edges have to be more equally distributed in G, and thus the contractions do not change the graph too much.

From a graph drawing perspective this approach offers a drawing such that all crossings happen in disjoint areas nearby the clusters and the rest of the meta-drawing is crossing-free. Such a meta-drawing resembles well the original graph and can be well grasped by a glance from the distance. The local crossings get inspected by taking a magnifying glass for particular clusters.

If $\ell = 1$, the problem is solvable in linear time as it becomes just planarity testing. For $\ell \geq 2$, we prove:

Structure. In Section 5.4, we describe our FPT algorithm for the P-RestrictedContract problem. In Section 5.7, we deal with the ℓ-SubContract problem. Last, in Section 5.8, we show how to simplify the proof of Grohe [35].

5.4 Restricted Contractibility is Fixed-Parameter Tractable

Let P be a fixed inclusion-closed and inert contraction-closed MSOL formula. In this section, we show that the problem P-RestrictedContract is fixed-parameter tractable with respect to the parameter k. Namely, we describe an algorithm which solves P-RestrictedContract in time $\mathcal{O}(n^2 \cdot f(k))$.

The basic structure of our algorithm is based on the following idea invented by Grohe [35]. If the graph has small tree-width, we can solve the problem by Courcelle Theorem [22]. If the tree-width is large, we find an embedded large hexagonal grid and produce a smaller graph to which we can apply the procedure recursively.

5.5 Definitions

We first introduce notation similar to that of Grohe's in [35].

Topological Embeddings. A topological embedding $h : G \hookrightarrow H$ of G into H consists of two mappings: $h_V : V(G) \to V(H)$ and $h_E : E(G) \to P(H)$, where $P(H)$ denotes paths in H. These mappings must satisfy the following properties:

- The mapping h_V is injective, distinct vertices of G are mapped to distinct vertices of H.
- For distinct edges e and f of G, the paths $h_E(e)$ and $h_E(f)$ are distinct, do not share internal vertices and share possibly at most one endpoint.
- If $e = uv$ is an edge of G then $h_V(u)$ and $h_V(v)$ are the endpoints of the path $h_E(e)$. If w is vertex of G different from u and v then path $h_E(e)$ does not contain the vertex $h_V(w)$.

For an example, see Fig. 5.4.

It is useful to notice that there exists a topological embedding $h : G \hookrightarrow H$, if there exists a subdivision of G which is a subgraph of H. For a subgraph $G' \subseteq G$, denote by $h \restriction G'$ the restriction of h to G'. For simplicity, we use the term *embeddings* instead of topological embeddings.

Hexagonal grid. We define recursively the hexagonal grid H_r of radius r (see Fig. 5.5). The graph H_1 is a hexagon (the cycle of length six). The graph H_{r+1} is obtained from H_r by adding $6r$ hexagonal faces around H_r as indicated in Fig. 5.5.

The nested *principal cycles* $C^1, \ldots, C^r$ are called the boundary cycles of $H_1, \ldots, H_r$. From the inductive construction of H_r, H_k is obtained from H_{k-1} by adding C^k and connecting it to C^{k-1}. A *principal subgrid* H_r^s where $s \leq r$ denotes the subgraph of H_r isomorphic to H_s and bounded by the principal cycle C^s of H_r.

Flat Topological Embeddings. Let H be a subgraph of a graph G. An *H-component* C of G is

- either a connected component of $G \setminus H$ together with the edges connecting C to H and its incident vertices, or
- an edge $e = uv$ and the incident vertices u and v such that $u, v \in V(H)$ and $e \notin E(H)$.

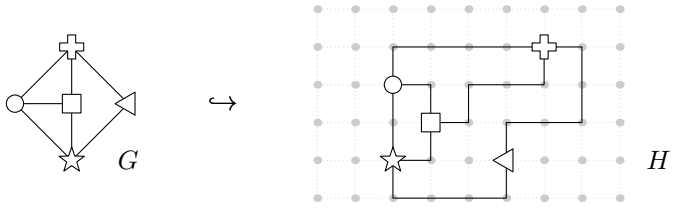


Figure 5.4: An example of a topological embedding $G \hookrightarrow H$.

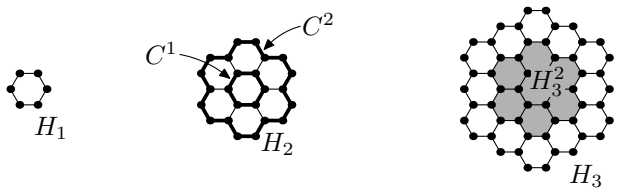


Figure 5.5: Hexagonal grids H_1, H_2 and H_3. Inside H_2, the principal cycles C^1 and C^2 are depicted in bold. Inside H_3, the principal subgrid H_3^2 is shown.

The endpoints of edges of C contained in H are called the *vertices of attachment* of C. Figure 5.6a illustrates the notion of *H-components*.

Let G be a graph and let $h : H_r \hookrightarrow G$ be an embedding of a hexagonal grid in G. An $h(H_r)$-component C is called *proper* if C has at least one vertex of attachment in $h(H_r) \setminus h(C^r)$, namely, the component is attached to an inner vertex of the grid. Let $h_+(H_r)$ denote the union of $h(H_r)$ with all proper $h(H_r)$-components. Notice that the proper $h(H_r)$-components may be obstructions to the planarity of $h_+(H_r)$. The embedding h is called a *flat embedding* if $h_+(H_r)$ is a planar graph. For an example, see Fig. 5.6b.

Tree-width. For a graph G, its tree-width is an integer k which describes how "similar" is G to a tree [23]. For our purposes, we use tree-width as a black box in our algorithm. The following two properties of tree-width are crucial.

Theorem 5.5.1 (Robertson and Seymour [59], Boadlaender [13], and Perkovič and Reed [57])**.** *For every $s \geq 1$, there is $t \geq 1$ and a linear-time algorithm*

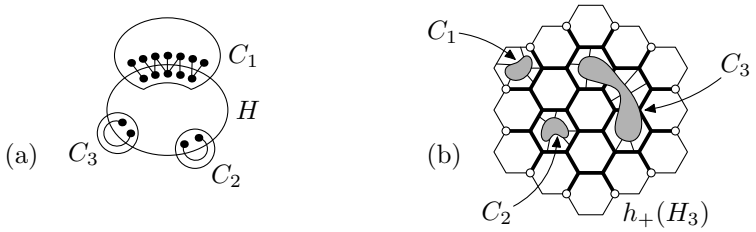


Figure 5.6: (a) H-components C_1, C_2 and C_3 of G. (b) A subgraph $h_+(H_3)$ consisting of $h(H_3)$ and three proper components C_1, C_2 and C_3 having attachments to inner vertices of $h(H_3)$ (highlighted in bold). The embedding h is not flat since C_3 obstructs planarity.

that, given a graph G, either (correctly) recognizes that the tree-width of G is at most t or returns an embedding $h : H_s \hookrightarrow G$.

Theorem 5.5.2 (Courcelle [22])**.** *For every graph G of tree-width at most t and every MSOL formula φ, there exists an algorithm that decides the formula φ on G in time $\mathcal{O}(n \cdot g(t))$, where n is the number of vertices of G.*

5.6 The algorithm

Overview. The general outline of the algorithm is as follows. It proceeds in two phases. The first phase deals with graphs of large tree-width and repeatedly modifies G to produce a graph of small tree-width. In addition, we keep a set $F \subseteq E$ of forbidden edges for contractions. Initially, F is empty and during the modification some edges are added. The second phase reduces P-RestrictedContract to solving an MSOL formula which is done by Courcelle's Theorem 5.5.2.

Phase I. We first prove the following lemma, which states that in an embedded large hexagonal grid H_s into G, we either find a flat hexagonal grid H_r (smaller than H_s), or else G is not k-contractible. This lemma represents the most significant difference from the paper of Grohe [35], as illustrated in Section 5.8.

Lemma 5.6.1. *Let G be a k-contractible graph. For every $r \geq 1$, there exists $s \geq 1$ such that for every embedding $h : H_s \hookrightarrow G$ there is some subgrid $H_r \subseteq H_s$ such that $h \restriction H_r$ is a flat embedding.*

Proof. For given r and k, we fix s and t large enough as follows: We choose $s \approx 2kt$ so that H_s contains $2k+1$ *disjoint subgrids* $H_{t_1}, \ldots, H_{t_{2k+1}}$ of radius t and let H'_{t_i}, formerly denoted by $H^{t-2}_{t_i}$, be a subgrid of H_{t_i} without two outermost layers. We choose $t \approx 7rk$ so that each H'_{t_i} contains $7k+1$ *disjoint subgrids* $H_{t_i,r_1}, \ldots, H_{t_i,r_{7k+1}}$ of radius r.[2] In this way, we get the hierarchy of nested subgrids as in Fig. 5.7:

$$H_s \supsetneq H_{t_i} \supsetneq H'_{t_i} \supsetneq H_{t_i,r_j}, \qquad \text{where } 1 \leq i \leq 2k+1, \quad 1 \leq j \leq 7k+1.$$

We next argue using the pigeon-hole principle that for some H_{t_i,r_j} is the embedding $h \restriction H_{t_i,r_j}$ a flat embedding.

[2] Actually just $s \approx \sqrt{2kt}$ and $t \approx \sqrt{7kr}$ would be sufficient.

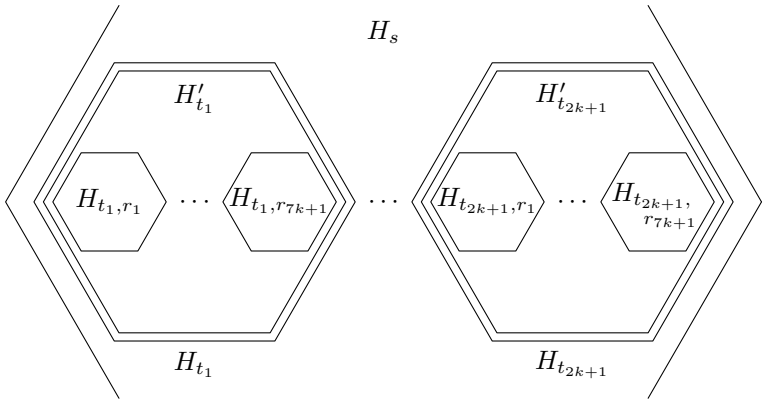


Figure 5.7: The hierarchy of the hexagonal grids nested in H_s.

Since we are assuming that G is k-contractible, we can fix a matching planarizing set S and consider one subgrid H_{t_i}. Let a *cell* be an h-image of a hexagon of H'_{t_i}. We call an $h(H'_{t_i})$-component *bad* if it contains an edge from S. A cell is considered *bad* if it contains an edge of S or if there is at least one bad $h(H'_{t_i})$-component attached to the cell. Since bad cells have some obstructions to planarity attached to them, we will exhibit some grid H_{t_i,r_j} such that its embedding $h(H_{t_i,r_j})$ avoids all bad cells.

To proceed, call an $h(H'_{t_i})$-component C *large* if the vertices of attachment of C are not contained in one cell. As an example, in Fig. 5.6b, the component C_3 is large, but C_1 and C_2 are not. Large $h(H'_{t_i})$-components posses the following useful properties which we prove afterwards in a series of claims.

1. If an $h(H'_{t_i})$-component is large, then we can embed $K_{3,3}$ into $h_+(H_{t_i})$. This means that there must be some H_{t_i} having no large $h(H'_{t_i})$-component, otherwise the graph would not be k-contractible.
2. On the other hand, if a bad $h(H'_{t_i})$-component is not large, it can produce at most seven bad cells. This implies that $h(H'_{t_i})$ must have a number of bad cells bounded by $7k$ and therefore for some j, the embedding $h(H_{t_i,r_j})$ is a flat embedding.

Claim 5.6.2. *Let C be a large $h(H'_{t_i})$-component. Then we can embed $K_{3,3}$ into $h_+(H_{t_i})$ such that $K^-_{3,3} := K_{3,3} - e$ is embedded into the grid $h(H_{t_i})$.*

Proof. Instead of a tedious formal proof, we illustrate the main idea in Fig. 5.8. If C is large, it has two vertices u and v not contained in one

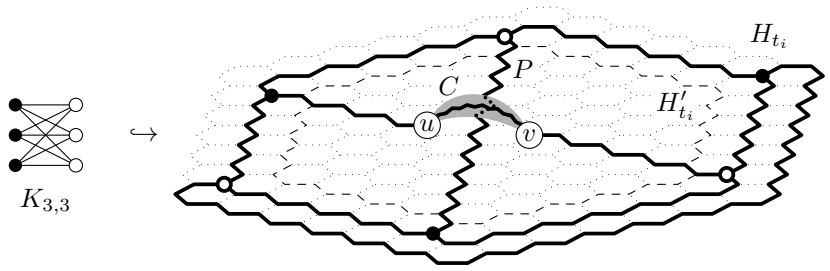


Figure 5.8: The component C acts as a bridge allowing two non-crossing paths across the grid. Thus we can embed $K_{3,3}$ into $h(H_{t_i})$.

cell. Thus there exists a path P going across the grid "between" u and v. Using C as a "bridge" from u to v, we can cross P by another path across the grid. These two paths together with two outer layers of $h(H_{t_i})$ allow to embed $K_{3,3}$ into $h_+(H_{t_i})$ such that $K^-_{3,3}$ is embedded into $h(H_{t_i})$. ◇ □

Claim 5.6.3. *There is some H_{t_i} such that there is no large $h(H'_{t_i})$-component.*

Proof. According to the above claim, if H_{t_ℓ} contains a large $h(H'_{t_\ell})$-component, we can embed $K_{3,3}$ into $h_+(H_{t_\ell})$. Since $G \circ S$ is a planar graph, this embedding of $K_{3,3}$ has to be contracted by S. To contract it, there has to be an edge $e \in S$ incident with $h(K^-_{3,3})$, otherwise $G \circ S$ still contains an embedding of $K_{3,3}$. Therefore e is incident with $h(H_{t_\ell})$. We know that $|S| \leq k$ and each edge in S is incident with at most two grids H_{t_ℓ}. Since we have $2k+1$ disjoint grids, there is some H_{t_i} not having any edge S incident with $h(H_{t_i})$.[3] Therefore, there is no large $h(H'_{t_i})$-component. ◇ □

Claim 5.6.4. *For this H_{t_i}, the embedding $h(H'_{t_i})$ contains at most $7k$ bad cells.*

Proof. Let $e \in S$. Each $h(H'_{t_i})$-component C is not large, so it is attached to at most seven cells. Therefore, if $e \in C$, we get at most seven bad cells. If e belongs to a cell directly, we get two bad cells. Since $|S| \leq k$, we get at most $7k$ bad cells. ◇ □

[3]By a more refined analysis, one can show that only $k+1$ disjoint grids suffice. The reason is that if an edge is incident to two grids, it contracts neither of the embeddings $h(K_{3,3})$.

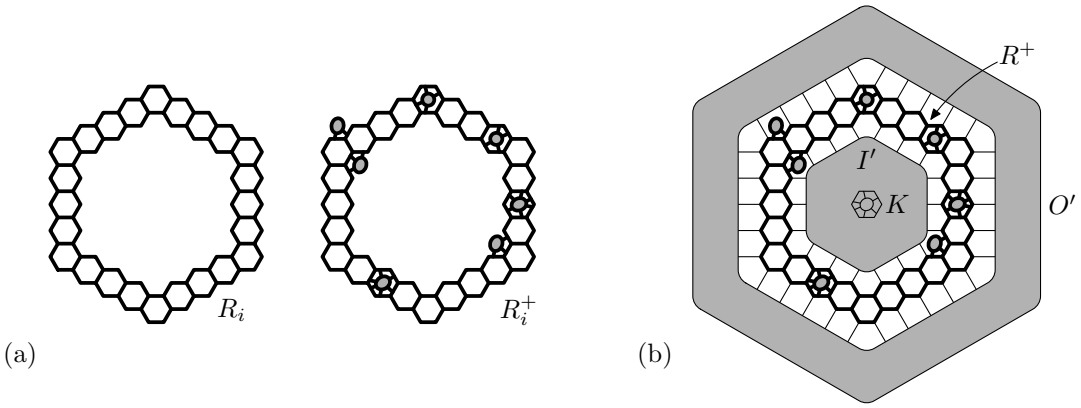


Figure 5.9: (a) An example of a ring R_i and an extended ring R_i^+. (b) The extended ring R^+ (in bold) splits $G' \setminus R$ into an outer part O' and an inner part I'. The inner part is connected and contains H_r^{2i-1} and especially K.

We have established these claims and we conclude the proof of Lemma 5.6.1. By the pigeon-hole principle, there exists one of $h(H_{t_i,r_1}), \ldots, h(H_{t_i,r_{7k+1}})$ containing no bad cells, and we denote it by H_{t_i,r_j}. We know that $h_+(H_{t_i,r_j})$ has no edges of S and it is planar; in other words $h(H_{t_i,r_j})$ is a flat embedding. □ □

The next lemma shows that a small part of $h_+(H_r)$ is never contracted by a minimal set S. Let the *kernel* K of $h_+(H_r)$ denote the h-image of the central principal cycle $h(C^1)$ together with the $h(H_r)$-components attached only to $h(C^1)$.

Lemma 5.6.5. *Let G be a k-contractible graph and let S be a minimal planarizing set of G. Let $h : H_r \hookrightarrow G$, where $r \geq 2k+3$, be a flat embedding and let K be the kernel of $h_+(H_r)$. Then the edges of G incident with the vertices of K do not belong to S.*

Proof. We define a *ring* R_i as $h(H_r^{2i+1} \setminus H_r^{2i-1})$, i.e., it is the h-image of the two consecutive principal cycles of H_r together with the edges between them. An *extended ring* R_i^+ is the union of R_i with all the $h(H_r)$-components having all the vertices of attachment in R_i. See Fig. 5.9a. Consider $2k+1$ disjoint extended rings $R_1^+, \ldots, R_{2k+1}^+$ and notice that for some i, no edge of S is incident with a vertex of R_i^+. Denote $R := R_i$, $R^+ := R_i^+$ and the principal cycles of this ring as $C_{\text{outer}} := h(C^{2i+1})$ and $C_{\text{inner}} := h(C^{2i})$.

The graph $G' := G \circ S$ is planar with a specific embedding Δ' as follows; see Fig. 5.9b. The ring R itself is a subdivision of a 3-connected graph, and therefore it has a unique embedding up to the choice of an outer-face. We choose a drawing having C_{outer} as the outer-face of $\Delta' \restriction R$. For the extended ring R^+, the embedding of the attached components is not unique, but they are attached somehow to R.

Now, $G' \setminus R$ is split by R to two parts: the *inner part* I' lying inside the face bounded by C_{inner} and the *outer part* O' inside the outer-face bounded by C_{outer}. Moreover, we can assume for Δ' that I' is connected, containing $h(H_r^{2i-1}) \circ S$. The reason is that if some part is disjoint from the grid, it either belongs to R^+, or it is a separate component of G', so we choose Δ' such that it is drawn into the outer-face.

Since contraction does not change connectivity in G and all contractions avoid R^+, the subgraph $G \setminus R^+$ is separated into parts I (containing $h(H_r^{2i-1})$ and especially K) and O (containing the rest) such that $I \circ S = I'$ and $O \circ S = O'$ with no edges between I and O. We show next that the minimality of S forces that $I \cap S = \emptyset$.

We take the embedding Δ', remove $\Delta' \restriction I'$ and replace it with some embedding of I. Since $h(H_r)$ is a flat embedding, the graph $h_+(H_r)$ is planar, so in particular the subgraph induced by $R^+ \cup I$ is planar. We consider one of its planar embeddings Δ having I embedded into the inner face of R^+. Since $\Delta \restriction I$ has the same orientation of edges to R as $\Delta' \restriction I'$, it is possible to replace the embedding of I' in Δ' by $\Delta \restriction I$. This means that it is not necessary to contract the edges of I and therefore $I \cap S = \emptyset$. The statement follows since K and its incident edges belong to I. □ □

We note that the algorithm just contracts K and not the whole I since we do not know which extended ring R_i^+ avoids edges of S.

Recall that F is the set of forbidden edges to contract. If the graph G

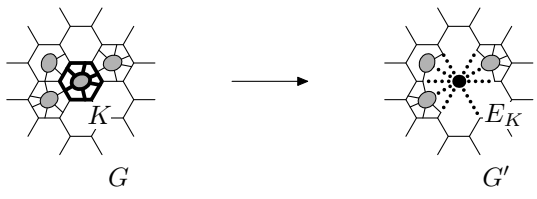


Figure 5.10: The graph is modified by contraction of the kernel K and adding its incident edges E_k to F.

is k-contractible by a planarizing set S such that $S \cap F = \emptyset$, we say that G is (k, F)-*contractible*. As we process our algorithm, we use Lemma 5.6.5 to modify the input G and F to a smaller graph G' and an extended set F' as follows. The kernel K is contracted into a single vertex, so $G' := G \circ K$, d the edges E_K between $V(K)$ and $V(G) \setminus V(K)$ are added to F, so $F' := F \cup E_K$. Figure 5.10 depicts this modification.

Lemma 5.6.6. *The graph G is (k, F)-contractible if and only if the graph G' is (k, F')-contractible.*

Proof. According to Lemma 5.6.5, a minimal planarizing set S for G avoiding F does not contain any edges of K and F'. Therefore, it is also a planarizing set of G' avoiding F'.

On the other hand, assume that G' has a planarizing set S disjoint from F' of size at most k. We want to show that S is also a planarizing set for G. We consider an embedding of $G' \circ S$ and we replace the vertex created by contracting the kernel K by an embedding of K in a manner completely analog to the one described in the proof of Lemma 5.6.5. We get a planar embedding of $G \circ S$. □ □

Phase II. We show next that when the tree-width of G is small, we can solve (k, F)-contractibility with respect to P using Courcelle's theorem [22]. To this effect all we need to show is that it is possible to express (k, F)-contractibility in the monadic second-order logic (MSOL).

Lemma 5.6.7. *For a fixed graph H, there exists an MSOL formula $\mu_H(S, G)$ which is satisfied if and only if $G' := G \circ S$ contains H as a minor.*

Proof. We modify a well-known formula $\widetilde{\mu}_H(G)$ for testing whether H is a minor of G. For $|H| = \ell$, the formula $\widetilde{\mu}_H(G)$ tests whether there exist disjoint sets of vertices $V_1, \ldots, V_\ell$ (representing the sets of vertices contracted to vertices of H) such that

- for every $v_i v_j \in E(H)$ there exists an edge between V_i and V_j in G, and
- each set V_i is connected in G.

Let $S = \{e_1, \ldots, e_k\}$ and $e_j = x_j y_j$. To test whether H is minor in G', we require that each V_i either contains both endpoints of e_j, or none of them. Formally,

$$\mu_H(S, G) = \widetilde{\mu}_H(G) \wedge \bigwedge_{\substack{1 \le i \le \ell \\ 1 \le j \le k}} (x_j \in V_i \iff y_j \in V_i).$$

□ □

Lemma 5.6.8. *There exists a formula $\varphi_k(F, G)$ which is satisfiable if and only if G is (k, F)-contractible with respect to the MSOL formula P.*

Proof. This formula is defined as follows:

$$\varphi_k(F, G) := \exists S \subseteq E(G) : |S| \leq k \wedge (S \cap \square F = \emptyset) \wedge P(S, G) \wedge \neg\mu_{K_5}(S, G) \wedge \neg\mu_{K_{3,3}}(S, G).$$

Putting all the Pieces Together. We finish this section with a proof of the announced Theorem 5.3.1, stating that P-RestrictedContract for an inclusion-closed and inert contraction-closed MSOL formula $P(S, G)$ is solvable in time $\mathcal{O}(n^2 \cdot f(k))$.

Theorem 5.3.1. See Algorithm 1 for a pseudocode; Phase I corresponds to the steps 3 to 7, Phase II to the step 8. Depending on k, we choose a suitable value for s so we can apply Lemmas 5.6.1 and 5.6.5. By Theorem 5.5.1, there is a corresponding value of t.

We repeat Phase I till tree-width of G becomes at most t. We find an embedding h of a large hexagonal grid H_s, by Theorem 5.5.1 in linear time. Using Lemma 5.6.1, we can find a subgrid H_r such that $h(H_r)$ is flat. Moreover, we can find such H_r in time $\mathcal{O}(k^2 n)$ by testing planarity for all $h_+(H_{t_i,r_j})$. We contract the kernel K and we modify the graph G and the set of forbidden edges F. Lemmas 5.6.5 and 5.6.6 show that this modification does not

Algorithm 1 P-RestrictedContract

Require: A graph G and an inclusion-closed and inert contraction-closed formula $P(S, G)$.

Ensure: A planarizing set S of size at most k satisfying $P(S, G)$ if it exists.

1: Initialize the set of forbidden edges $F := \emptyset$.
2: Depending on k, choose suitable $s \geq 1$ and $t \geq 1$ for Theorem 5.5.1.
3: **while** the tree-width of G is bigger than t **do**
4: Find an embedding $h : H_s \hookrightarrow G$ using Theorem 5.5.1.
5: Find a subgrid H_r such that $h \restriction H_r$ is a flat embedding as described in Lemma 5.6.1.
6: Modify the graph as $G := G \circ K$ and $F := F \cup E_K$.
7: **end while**
8: **return** a planarizing set S satisfying $\varphi_k(F, G)$ using Theorem 5.5.2 if it exists.

change solvability of the problem. After each modification, we get a smaller graph G. Therefore we need to repeat this at most $\mathcal{O}(n)$ times, so the total running time of Phase I is $\mathcal{O}(n^2 \cdot p(k))$ for some function p.

Let G denote the original graph and let G' denote the modified graph created by Phase I. Phase II uses Theorem 5.5.2 to solve the MSOL formula $\varphi_k(F', G')$ in time $\mathcal{O}(n \cdot q(k))$. By Lemmas 5.6.1, 5.6.5 and 5.6.6, the modified graph G' is (k, F') contractible if and only if the original graph is k-contractible. It remains to show that the modification is not changing solvability of $P(S, G)$. Since $P(S, G)$ is inclusion closed, we can concentrate on inclusion minimal planarizing sets in G and G'. Further by Lemma 5.6.5, each modification contracts some inert edges which are not incident with any inclusion minimal planarizing set S. Since $P(S, G)$ is inert contraction-closed, each modification does not change solvability of $P(G, S)$. So testing $\varphi_k(F', G')$ correctly tests whether G is k-contractible with respect to $P(S, G)$.

The overall complexity of the algorithm is $\mathcal{O}(n^2 \cdot f(k))$ for some function f. □ □

5.7 ℓ-subgraph Contractibility

We first show that for a fixed ℓ, testing ℓ-subgraph contractibility can be done in time $\mathcal{O}(n^2 \cdot f'(k))$ for some function f'. It would follows from Theorem 5.3.1 and the fact that ℓ-subgraph contractibility is expressible using MSOL the following corollary.

Corollary 5.7.1. *For every fixed ℓ, the problem ℓ-*SubContract *can be solved in time $\mathcal{O}(n^2 f'(k))$ where n is the number of vertices.*

Proof. We just describe in words how to construct the MSOL formula P and we check that P is inclusion-closed and inert contraction-closed. The length of the formula may depend on k and ℓ. The formula P tests whether there exists a decomposition of the edges in S into pairwise disjoint sets $E_1, \ldots, E_k$ satisfying the following properties. Let $V_1, \ldots, V_k$ denote the corresponding sets of vertices incident with $E_1, \ldots, E_k$, respectively. The formula P is satisfied if and only if $|V_i| \leq \ell$ for each i and the sets $V_1, \ldots, V_k$ are pairwise disjoint. This solves ℓ-subgraph contractibility and clearly is an inclusion-closed MSOL formula.

It remains to show that P is inert contraction-closed. Assume that $P(S, G)$ is solvable and let S be any inclusion minimal solution. For ev-

ery inert set B, no edge of B is by the definition incident with an edge of S. Therefore S is a planarizing set of $G \circ B$ consisting of ℓ-subgraphs and $P(S, G \circ B)$ is satisfiable. For the other implication, if S is an inclusion minimal solution of $P(S, G \circ B)$, then it also solves $P(S, G)$. The reason is that B is inert, so no edge of S is incident with an edge of B. Therefore, S is a planarizing set of G consisting of ℓ-subgraphs.

Since ℓ-subgraph contractibility can be expressed using MSOL formulas which is an inclusion-closed and P inert contraction closed we can apply Theorem 5.3.1. □ □

Matching Contractibility. In the rest of this section, we establish NP-completeness of ℓ-SubContract. To do so, we first introduce a new problem called *matching contractibility*. The graph G is *F-matching contractible* with respect to a set of edges F if there exists a planarizing set S which forms a matching in G and $S \cap F = \emptyset$.

Problem:	MatchingContract
Input:	An undirected graph G and a set of forbidden edges $F \subseteq E$.
Output:	Is G an F-matching contractible graph?

First, we show that ℓ-subgraph contractibility can solve matching contractibility.

Lemma 5.7.2. *Matching contractibility is reducible from ℓ-subgraph contractibility.*

Proof. For an input G and F, we produce a graph G' which is ℓ-subgraph contractible if and only if G is F-matching contractible. We replace the edges of G by paths:

- if $e \in F$, then we replace it by a path of length ℓ, and
- if $e \notin F$, the we replace it by a path of length $\ell - 1$.

Also, we put $k = |E(G')|$ so only ℓ-subgraphs restrict S.

If a planarizing set S is a matching in G avoiding F, then we can contract the corresponding paths in G' by ℓ-subgraphs. On the other hand, let S' be a planarizing set of G' consisting of ℓ-subgraphs. First, we ignore ℓ-subgraphs not contracting entire paths, since their contraction preserves the

topological structure of the graph. If a path in G' corresponding to $e \in E(G)$ is contracted, it has to be contracted by a single ℓ-subgraph. In such a case, $e \notin F$, otherwise the path is too long. Also, the contracted paths have to form a matching since the ℓ-subgraphs cannot share the end-vertices belonging to G. So the planarizing set S' of G' gives a planarizing set S of G which is a matching and which avoids F. □ □

Overview of the Reduction. To show NP-hardness of MatchingContract, we present a reduction from Clause-Linked Planar 3-SAT. An instance I of Clause-Linked Planar 3-SAT is a Boolean formula in CNF such that each variable occurs in exactly three clauses, once negated and twice positive, each clause contains two or three literals and the incidence graph of I is planar. Fellows et al. [25] show that this problem is NP-complete.

Given a formula I, we construct a graph G_I with a set F_I of forbidden edges such that G_I is F_I-matching contractible if and only if I is satisfiable. The construction is performed by replacing each variable x by a variable gadget G_x, and replacing each clause c by a clause gadget H_c. All variable gadgets G_x are isomorphic and we have two types of clause gadgets H_c, depending on the size of c. These gadgets consist of several copies of the graph K_5 with most of the edges in F_I. In Fig. 5.11, the edges not contained in F_I are represented by bold lines. Each variable gadget contains three pendant edges that are identified with certain edges of the clause gadgets, thus connecting the variable and clause gadgets.

Variable Gadget. Let x be a variable which occurs positive in clauses c_1 and c_2, and negative in a clause c_3. The corresponding variable gadget G_x is depicted in Fig. 5.11a. It consists of four copies of K_5, each having all but two edges in F_I. Three of the copies of K_5 have pendant edges attached, denoted by

$$e(x, c_i) = v(x, c_i)w(c_i), \qquad i \in \{1, 2, 3\};$$

refer to Fig. 5.11a. These edges also belong to the clause gadgets H_{c_1}, H_{c_2} and H_{c_3}. All other vertices and edges are private to the variable gadget G_x.

The main idea behind the variable gadget is that exactly one of the edges t_x and f_x is contracted. This encodes the assignment of the variable x as follows: the edge t_x is contracted for true and f_x for false. The edges $e(x, c_1)$ and $e(x, c_2)$ shared with the clause gadgets H_{c_1} and H_{c_2}, can be contracted only if t_x is contracted, and $e(x, c_3)$ shared with H_{c_3} can be contracted if and only if f_x is contracted.

Clause Gadget. Let c be a clause containing variables x, y and possibly z. The clause gadget H_c is the graph K_5 with all but 2 or 3 edges in F_I. The edges that are not forbidden to contract share a common vertex $w(c)$ and they are the edges $e(x,c)$, $e(y,c)$ and possibly $e(z,c)$ shared with the variable gadgets G_x, G_y and possibly G_z; see Fig. 5.11b.

To make the clause gadget planar, we need to contract exactly one of the edges $e(x,c)$, $e(y,c)$ and possibly $e(z,c)$. This is possible only if the clause is satisfied by the corresponding variable evaluated as true in this clause.

Lemma 5.7.3. *The graph G_I is F_I-matching contractible if and only if I is satisfiable.*

Proof. $\Longrightarrow$: Suppose first that G_I is F_I-matching contractible, and let $S \subseteq E(G_I)$ be a matching planarizing set. Using S, we construct a satisfying assignment of I. Consider a variable x. In G_x, each copy of K_5 needs to have at least one edge contracted by S.

Exactly one of t_x and f_x is in S. If $t_x \in S$, then t'_x cannot be in S (note that S is a matching), hence $e'(x,c_3) \in S$, and $e(x,c_3)$ cannot be in S. On the other hand, if $t_x \notin S$, necessarily $f_x \in S$, and by a similar sequence of arguments, none of $e(x,c_1)$ and $e(x,c_2)$ is in S.

We define a truth assignment for the variables of I so that x is true if and only if $t_x \in S$. It follows that if x is evaluated as false in a clause c,

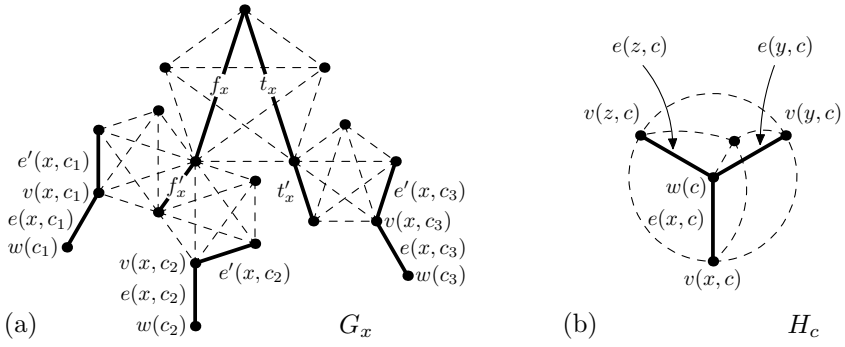


Figure 5.11: The bold edges can be contracted and the dashed edges are forbidden edges from F_I. (a) The variable gadget G_x where the three outgoing edges are shared with clause gadgets H_{c_1}, H_{c_2} and H_{c_3}. (b) The clause gadget H_c. The edge $e(z,c)$ may also be in F_I if the clause contains only two variables x and y. The two or three contractible edges are shared with variable gadgets G_x, G_y and possibly G_z.

then the edge $e(x, c)$ is not in S. Since S contains exactly one edge of H_c, in each clause gadget at least one variable must be evaluated to true. Thus I is satisfiable.

$\Longleftarrow$: Suppose that I is satisfiable and fix a satisfying truth assignment ϕ. We set

$$S = \{t_x, f'_x, e(x, c_1), e(x, c_2), e'(x, c_3) \mid \phi(x) = \text{true}\} \cup \tag{5.1}$$
$$\{f_x, t'_x, e'(x, c_1), e'(x, c_2), e(x, c_3) \mid \phi(x) = \text{false}\}. \tag{5.2}$$

For the variable gadgets, the set S is a matching. Each clause gadget contains at least one edge of S. But if a clause, say c, contains more than one variable evaluated as true, then its clause gadget H_c contains in S more edges with the common vertex $w(c)$. In such a case, we perform a *pruning operation* on H_c, i.e, remove all edges from $S \cap E(H_c)$ but one. The resulting set S' is a matching such that each K_5 in G_I contains exactly one edge of S'.

It only remains to argue that this set S' is a planarizing set. The graph $G' = G \circ S'$ consists of copies of K_4 glued together by vertices or edges. Each copy is attached to other copies by at most three vertices. Since K_4 itself has a non-crossing drawing in the plane such that three of its vertices lie on the boundary of the outer face, and these vertices can be chosen arbitrarily as well as their cyclic order on the outer face, the drawings of the variable and clause gadgets can be combined together along a planar drawing of the incidence graph of I. Therefore, G' is planar. □ □

We can finish prove of the following:

Theorem 5.7.4. *For $\ell \geq 2$, the problem ℓ-SubContract is NP-complete.*

Proof. Clearly, ℓ-SubContract belongs to NP. By Lemma 5.7.3 and [25], the problem MatchingContract is NP-hard. Lemma 5.7.2 implies that ℓ-SubContract is NP-hard. □ □

We note that the problem ℓ-SubContract remains NP-complete when generalized to surfaces of a fixed genus g (instead of planar graphs). Consider a graph H_g such that for every embedding of H_g into the surface, each face is homeomorphic to the disk. We modify our reduction by taking $G_I \cup H_g$ as the graph and by adding all the edges of H_g into F. For each surface, there exists such a graph H_g (see triangulated surfaces in Mohar and Thomassen [53]).

5.8 Simplifying Grohe's Approach

Grohe [35] gives an FPT algorithm for computing the crossing number k of a graph, in time $\mathcal{O}(n^2 f(k))$. Our FPT algorithm is based on his approach. On the other hand, we can simplify his argument in a similar manner as in the proof of Lemma 5.6.1. We describe this simplification here.

Grohe uses Thomassen's Theorem [66] which states the following:

Theorem 5.8.1 (Thomassen [66])**.** *Let G be a graph of genus at most k. For every $r \geq 1$, there is $s \geq 1$ such that for every topological embedding $h : H_s \to G$, there exists a subgrid $H_r \subseteq H_s$ such that the restriction $h \restriction H_r$ of h is flat.*

This result can be used since the crossing number is upper bounded by the genus, so it works as Lemma 5.6.1 in our FPT algorithm. With our simplification, we can completely avoid the notion of genus which is independent of the notion of crossing number. Therefore, we do not need to consider the more complicated theory of graphs on surfaces. Also, our proof is quite short and elementary.

Consider a plane drawing D of a graph G. The *facial distance* $d(x, y)$ of vertices x, y in G is the minimal number of intersections of D and a simple curve $\mathcal{C}$ connecting x and y, where $\mathcal{C}$ avoids the vertices of G. We use Riskin's theorem [58]:

Theorem 5.8.2 (Riskin [58])**.** *If G is a 3-connected cubic planar graph, then*

$$\mathrm{cr}(G + xy) = d(x, y),$$

where $G + xy$ is the graph G with the added edge xy.

Lemma 5.8.3. *Let G be a graph with $\mathrm{cr}(G) \leq k$. For every $r \geq 1$, there is $s \geq 1$ such that for every topological embedding $h : H_s \to G$, there exists a subgrid $H_r \subseteq H_s$ such that the restriction $h \restriction H_r$ of h is flat.*

Proof. Similarly as in Lemma 5.6.1, we define a hierarchy of grids as in Fig. 5.12. We choose $s \approx (k + r) \cdot k$ so that H_s contains $k + 1$ *disjoint subgrids* $H_{t_1}, \ldots, H_{t_{k+1}}$ of radius $k + r$. Let H'_{t_i} denote the principal subgrid $H^r_{t_i}$ of H_{t_i}. In this way, we get the hierarchy:

$$H_s \supsetneq H_{t_i} \supsetneq H'_{t_i}, \qquad \text{where } 1 \leq i \leq k + 1.$$

We want to show using the pigeon-hole principle that for some H'_{t_i} is the embedding $h \restriction H'_{t_i}$ a flat embedding.

Claim 5.8.4. *Let C be a proper $h(H'_{t_i})$-component. Then C has no attachment points in $h(H_s) \setminus h(H_{t_i})$.*

Proof. For contradiction, let x be an inner vertex of $h(H'_{t_i})$ which is a point of attachment of C, and let y be a point of attachment of C in $h(H_s) \setminus h(H_{t_i})$. Then there exists a path P from x to y with no internal vertices in $h(H_s)$. Consider $h(H_s)$ together with P and apply Riskin's Theorem 5.8.2. (We note that subdividing a graph preserves the crossing number.) Since the face distance $d(x, y)$ is at least $k + 1$, we get that $\mathrm{cr}(G) > k$ which is a contradiction. ◇ □

It follows from the claim that the graphs $h_+(H'_{t_i})$ are pairwise disjoint. Since $\mathrm{cr}(G) \leq k$, at least one of them is by the pigeon-hole principle planar. So the embedding $h(H'_{t_i})$ is flat for some H'_{t_i}. □ □

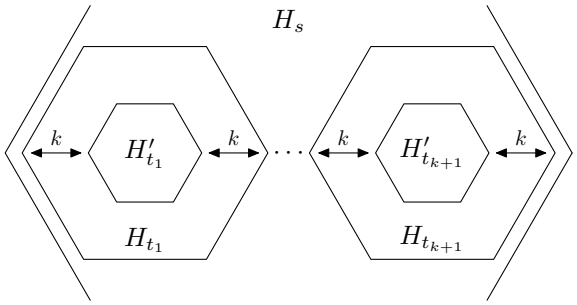


Figure 5.12: The hierarchy of the hexagonal grids nested in H_s.

Conclusion and Open Problems

Despite the fact that representations of graphs have been studied for years, there are several open problems and problems solved just partially. The community of Graph Drawing is very much interested in tractability of problems which are in general NP-hard and also in more effective algorithms for problems which are already polynomially tractable. The following is a list of open problems closely related to the topics studied in this thesis.

- *Does there exist a linear time fixed parameter algorithm which decides contractibility of graph?* We show in Chapter 5 a quadratic algorithm solving contractibility where we use the approach of Grohe [35]. But similarly to the case of crossing numbers [42], there might be a linear time algorithm.

- *Is every planar graph contained in B_1-VPG?* This conjecture is from [21] by Chaplick and Ueckerdt who showed that every planar graph is contained in B_2-VPG. This conjecture seems to have a valid adjustment because it was checked by computer that all planar graphs on at most ten vertices are intersection graphs of L-shapes[4] [30].

- *Is every planar graph contained in* {L, Γ}*-graphs?* This is strengthening of the previously mentioned conjecture with an interesting consequence. This particular conjecture was asked by Felsner et al. [26]. It is appealing even from another perspective, because if true, it would provide an alternative proof to Scheinerman's conjecture, since Middendorf and Pfeiffer [52] showed that every {L, Γ}-representation has a combinatorially equivalent SEG-representation.

[4] which is the same class as B_1-VPG

- *Is every planar graph representable by an intersection graph of segments in only 4 directions?* If this conjecture from [69] (in a stronger form) gets proven it would mean an alternative proof to the 4-color theorem.

Bibliography

[1] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787, 2008.

[2] Noga Alon, Asaf Shapira, and Benny Sudakov. Additive approximation for edge-deletion problems. *Automata, Languages and Programming*, 4051:1–2, 2006.

[3] Christoph Ambühl and Uli Wagner. The clique problem in intersection graphs of ellipses and triangles. *Theory of Computing Systems*, 38(3):279–292, 2005.

[4] MHS Amin, Peter J Love, and CJS Truncik. Thermally assisted adiabatic quantum computation. *Physical review letters*, 100(6):060503, 2008.

[5] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[6] Takao Asano and Tomio Hirata. Edge-deletion and edge-contraction problems. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 245–254, 1982.

[7] Andrei Asinowski, Elad Cohen, Martin Charles Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. String graphs of k-bend paths on a grid. *Electronic Notes in Discrete Mathematics*, 37:141–146, 2011.

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

[9] Jørgen Bang-Jensen, Bruce Reed, Mathias Schacht, Robert Šámal, Bjarne Toft, and Uli Wagner. On six problems posed by jarik nešetřil. In Martin Klazar, Jan Kratochvíl, Martin Loebl, Jiří Matoušek, Pavel Valtr, and Robin Thomas, editors, *Topics in Discrete Mathematics*, volume 26 of *Algorithms and Combinatorics*, pages 613–627. Springer Berlin Heidelberg, 2006.

[10] S Bellantoni, I. Ben-Arroyo Hartman, T Przytycka, and Sue Whitesides. Grid intersection graphs and boxicity. *Discrete mathematics*, 114(1):41–49, 1993.

[11] Rémy Belmonte, Petr A. Golovach, Pinar Heggernes, Pim van't Hof, Marcin Kamiński, and Daniël Paulusma. Finding contractions and induced minors in chordal graphs via disjoint paths. In Takao Asano, Shin-ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation*, volume 7074 of *Lecture Notes in Computer Science*, pages 110–119. 2011.

[12] Therese Biedl and Martin Derka. 1-string b_2-vpg representation of planar graphs. *arXiv preprint arXiv:1411.7277*, 2014.

[13] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small tree-width. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[14] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[15] 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.

[16] Martin L. Brady and Majid Sarrafzadeh. Stretching a knock-knee layout for multilayer wiring. *IEEE Transactions on Computers*, 39(1):148–151, 1990.

[17] Heinz Breu and David G Kirkpatrick. Unit disk graph recognition is np-hard. *Computational Geometry*, 9(1):3–24, 1998.

[18] A.E. Brouwer and H.J. Veldman. Contractibility and np-completeness. *Journal of Graph Theory*, 11(1):71–79, 1987.

[19] Jérémie Chalopin, Daniel Gonçalves, and Pascal Ochem. Planar graphs are in 1-string. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 609–617. Society for Industrial and Applied Mathematics, 2007.

[20] Jérémie Chalopin and Daniel 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, 2009.

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

[22] Bruno Courcelle. The monadic second-order logic of graphs III: Tree-decompositions, minors and complexity issues. *Informatique théorique et applications*, 26(3):257–286, 1992.

[23] Reinhard Diestel. *Graph theory*. Graduate Texts in Mathematics. 2005.

[24] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[25] Michael R. Fellows, Jan Kratochvíl, Martin Middendorf, and Frank Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13(3):266–282, 1995.

[26] Stefan Felsner, Kolja Knauer, George B. Mertzios, and Torsten Ueckerdt. Intersection graphs of l-shapes and segments in the plane. In *Mathematical Foundations of Computer Science 2014*, pages 299–310. Springer, 2014.

[27] Mathew C. Francis, Jan Kratochvíl, and Tomáš Vyskočil. Segment representation of a subclass of co-planar graphs. *Discrete Mathematics*, 312(10):1815–1818, 2012.

[28] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[29] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[30] Tomáš Gavenčiak. personal communication, 2015.

[31] R. Geldmacher and P. Liu. On the deletion of non-planar edges of a graph. In *In In Proceedings of the 10th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 727–738, 1977.

[32] Petr A. Golovach, Marcin Kamiński, and Daniël Paulusma. Contracting a chordal graph to a split graph or a tree. In *Mathematical Foundations of Computer Science 2011*, volume 6907 of *Lecture Notes in Computer Science*, pages 339–350. 2011.

[33] Petr A. Golovach, Marcin Kamiński, Daniël Paulusma, and DimitriosM. Thilikos. Increasing the minimum degree of a graph by contractions. In *Parameterized and Exact Computation*, volume 7112 of *Lecture Notes in Computer Science*, pages 67–79. 2012.

[34] Petr A. Golovach, Pim van't Hof, and Daniël Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013.

[35] Martin Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004.

[36] Sylvain Guillemot and Dániel Marx. A faster fpt algorithm for bipartite contraction. *Inf. Process. Lett.*, 113(22-24):906–912, 2013.

[37] Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1):59–84, 2000.

[38] Pinar Heggernes, Pim van't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. 13:217–228, 2011.

[39] Pinar Heggernes, Pim van't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. In *Parameterized and Exact Computation*, volume 7112 of *Lecture Notes in Computer Science*, pages 55–66. 2012.

[40] Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[41] Marcin Kamiński, Daniël Paulusma, and Dimitrios M. Thilikos. Contractions of planar graphs in polynomial time. In *Algorithms ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 122–133. 2010.

[42] Ken-ichi Kawarabayashi and Buce Reed. Computing crossing number in linear time. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 382–390. ACM, 2007.

[43] Jan Kratochvíl. String graphs. ii. recognizing string graphs is np-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 1991.

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

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

[46] Jan Kratochvíl and Aleš Kuběna. On intersection representations of co-planar graphs. *Discrete Mathematics*, 178(1-3):251–255, 1998.

[47] Jan Kratochvíl and Jiří Matoušek. String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B*, 53(1):1–4, 1991.

[48] 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.

[49] Asaf Levin, Daniel Paulusma, and Gerhard J. Woeginger. The computational complexity of graph contractions i: Polynomially solvable and np-complete cases. *Networks*, 51(3):178–189, 2008.

[50] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[51] George B. Mertzios. Graph modification problems. *Report from Dagstuhl Seminar 14071*, page 55, 2014.

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

[53] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001.

[54] Paul Molitor. A survey on wiring. *Journal of Information Processing and Cybernetics*, 27(1):3–19, 1991.

[55] Jan Kratochvíl; Jaroslav Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. 31(1):85–93, 1990.

[56] János Pach and Géza Tóth. Recognizing string graphs is decidable. In *Graph Drawing*, pages 247–260. Springer, 2002.

[57] Ljubomir Perković and Bruce Reed. An improved algorithm for finding tree decompositions of small width. *International Journal of Foundations of Computer Science*, 11(03):365–371, 2000.

[58] A. Riskin. The crossing number of a cubic plane polyhedral map plus an edge. *Studia Scientiarum Mathematicarum Hungarica*, 31(4):405–414, 1996.

[59] N. Robertson and P. D. Seymour. Graph minors. viii. a kuratowski theorem for general surfaces. *J. Comb. Theory Ser. B*, 48(2):255–288, 1990.

[60] Neil Robertson and Paul D. Seymour. Graph minors. xxii. irrelevant vertices in linkage problems. *J. Comb. Theory, Ser. B*, pages 530–563, 2012.

[61] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in np. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

[62] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in np. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

[63] Marcus Schaefer and Daniel Stefankovic. Decidability of string graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 241–246. ACM, 2001.

[64] E. R. Scheinerman. *Intersection classes and multiple intersection parameters of graphs*. PhD thesis, Princeton University, 1984.

[65] F. W. Sinden. Topology of thin film rc circuits. *Bell System Technical Journal*, 45(9):1639–1662, 1966.

[66] Carsten Thomassen. A simpler proof of the excluded minor theorem for higher surfaces. *Journal of Combinatorial Theory, Series B*, 70(2):306–311, 1997.

[67] Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981.

[68] Michael S Waterman and Jerrold R Griggs. Interval graphs and maps of dna. *Bulletin of Mathematical Biology*, 48(2):189–195, 1986.

[69] Douglas B. West. Open problems. *SIAM Journal on Discrete Mathematics, Newsletter 2*, (1):10–12, 1991.

[70] Mihalis Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 253–264. ACM, 1978.

[71] Mihalis Yannakakis. The effect of a connectivity requirement on the complexity of maximum subgraph problems. *J. ACM*, 26(4):618–630, 1979.