Algorithmic and Combinatorial Aspects of Containment Relations in Graphs

Rémy Belmonte

Dissertation for the degree of Philosophiae Doctor (PhD)

Department of Informatics
University of Bergen

2013

Acknowledgements

First and foremost, I would like to thank my PhD advisor, Pinar Heggernes, for all the help and support she has provided me with for the past three years. I am particularly thankful for all the advice (including several rounds of proof-reading of this thesis) and freedom I was given throughout these years, as well as the many opportunities to travel to various conferences and workshops, and to visit other great researchers. I would also like to thank Pim van 't Hof for roughing me up whenever I was relaxing a bit too much. I cannot emphasize enough how important both of them have been to the completion of this thesis. Most (if not all) of the people I have collaborated with throughout my PhD were met through either Pim or Pinar, and without them this thesis would (probably) not have existed. I am greatly indebted to both of you, thank you very much!

I want to thank my evaluation committee, Professors Vangelis Paschos, Peter Widmayer and Trond Steihaug, for accepting to review and evaluate my thesis.

I would also like to thank those without whom I wouldn't have gone to Bergen in the first place. I would like to thank Christophe Paul and Jan Arne Telle for arranging my first visit back when I was a Master's student, and Martin Vatshelle for helping me with the research for my Master's thesis.

I also want to thank Sang-Il Oum, Marcin Kamiński, Dimitrios Thilikos, Christophe Paul and Naoki Katoh for hosting me during research visits in Daejeon, Brussels and Warsaw, Athens, Montpellier and Kyoto respectively. I am especially grateful to Dimitrios for hosting me throughout the fall semester of 2012 and showing me around in Athens (although now that I think about it, we almost always ended up in Exarcheia and Thanasis), and giving me the opportunity to teach a few graph theory classes at the MPLA.

I am also greatly indebted to all of my coauthors: Archontia Giannopoulou, Petr Golovach, Pinar Heggernes, Pim van 't Hof, Marcin

Kamiński, Daniel Lokshtanov, Daniël Paulusma, Reza Saei, Dimitrios Thilikos and Martin Vatshelle.

One of the many great aspects of doing a PhD in Bergen is how lively and dynamic the Algorithms research group is. Everybody has my most sincere gratitude for making such a great working environment for the past three years. I would particularly like to thank Jesper Nederlof and Markus Dregi, my former and present officemates, Sigve Sæther and Pål Drange for the table tennis and chess games, as well as my other fellow PhD students Michał Pilipczuk and Sadia Sharmin.

My PhD studies were financed by the Norwegian Research Council through the project SCOPE, and I am grateful for this opportunity. In particular, I am very happy that I was able to do many interesting and highly useful travels and stays, thanks to this project.

Finally, I would like to thank my parents, sisters, extended family and friends, in particular for driving me around whenever I went back to France. Please try not to get too obviously bored during my defense, for those of you who will attend.

Bergen, August 23, 2013
Rémy Belmonte

Contents

Part I

Background

Chapter 1

Introduction

The problem of recognizing shapes, motifs and patterns arises in everyday life. An interesting example of this type of problem comes from the popular children's comic book series "Where is Waldo?", where the reader has to look at a picture in order to find a specific character among a sea of other, similar looking characters, as shown on Figure 1.1. While it may appear simplistic at first glance, as it is destined to be a children's game, this example of finding a cartoon character in a picture becomes harder as the picture grows in size. Fortunately, increasing the size of the picture does not drastically increase the number of characters it contains: if the picture gets twice as big as it used to be, the number of characters that are shown in the picture will typically also increase twofold. From this, and assuming we can find a new character and decide whether this character is indeed Waldo or not in constant time, e.g., at most a few seconds or so, we can conclude that Waldo can be found in time linear in the size of the picture given to the reader.

Considering the power of the notion of finding shapes/motifs/patterns, and the potential wealth of problems it encompasses, it is only natural that this idea has been adapted to more complex combinatorial structures, and graphs are undoubtedly one of the most natural such structures. Graphs form a simple and natural way to represent binary relations, and as such they have been extensively studied, both from a computational and a combinatorial point of view. These aspects give rise to two different families of problems:

- From a computational perspective, we can ask how fast one can find a given small "target" graph in a typically much larger "host" graph. Maybe one of the most common approaches is to ask whether there exist algorithms that achieve a certain type of running-time, e.g.,

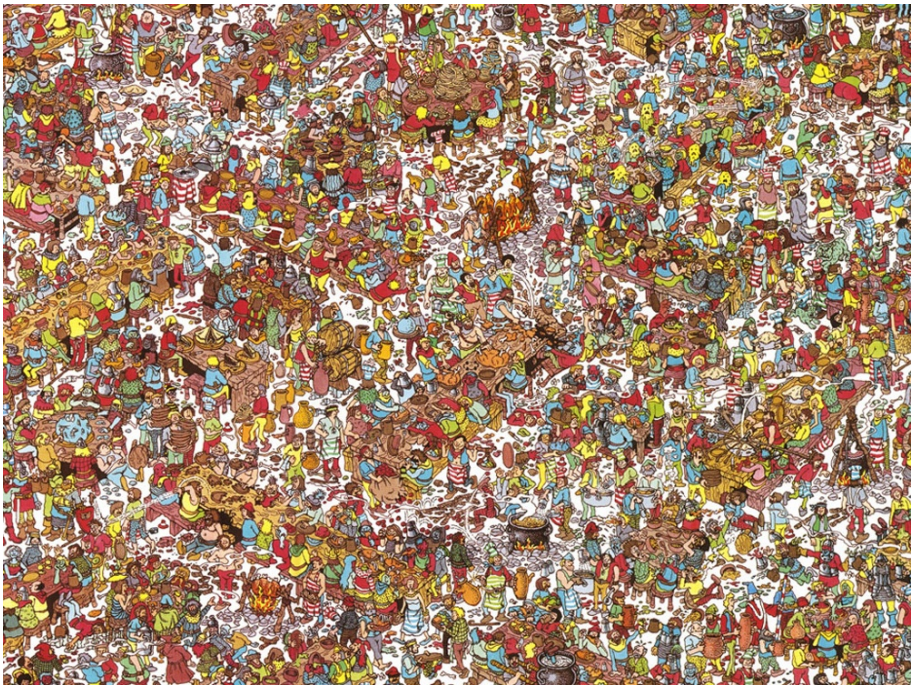


Figure 1.1: Where is Waldo? - The Great Waldo Search [81]

polynomial in the size of input.

- From a structural perspective, we can ask what kind of structure a graph must have in order not to contain a given graph as a pattern. In particular, is common to ask if the graphs that do not contain a certain target graph can be built in a certain way, starting from elementary "prime" graphs, and combining them by applying some predefined operations.

In this thesis, we study containment relations in graphs. It is of primary importance to note that the idea of a big graph "containing" a smaller one raises a natural question: what is the meaning of containment in the context of graph theory? The notion of containment in graphs and in many other combinatorial structures is very general. We will provide a formal definition of the notion of containment relation for graphs in Chapter 3, but here we want to give a non-technical description of the rules that we want a containment relation to follow. First, it is easy to argue that a containment relation should take as input an ordered pair of graphs, and output "yes" if the first graph is contained in the second one,

and "no" otherwise. In the context of Where is Waldo, this corresponds to the fact that the reader is given a pair of pictures: one of Waldo and the big picture in which the reader must find him. Additionally, we want the containment relation to satisfy the following three properties: reflexivity (every graph must contain itself), antisymmetry (if two graphs contain each other, then they must be the same graph) and transitivity (for every three graphs, if the first contains the second and the second contains the third, then the first must contain the third). This definition may seem rather abstract and arbitrary at first glance, this is why we spend the next section providing examples of famous graph theoretic problems that fit within this definition, as well as connections to other well known families of problems.

1.1 Examples of graph containment and related problems

In this section, we provide examples of classical problems in algorithmic graph theory that are containment problems. We also later discuss the relation between containment problems with some other families of problems such as so-called graph modification problems.

Probably some of the most natural examples of containment problems include problems such as finding a set of vertices of given size such that these vertices are all pairwise adjacent, i.e., finding a clique in a graph. If we ask the vertices to be all pairwise non-adjacent, then the problem becomes equivalent to finding an independent set of a given size. Another classical problem is to ask whether a graph contains a path that goes through every vertex of the graph exactly once. The same question can be asked when we replace path with cycle, and these two problems are equivalent to finding Hamiltonian path or a Hamiltonian cycle respectively.

It is also interesting to observe that, beyond these first examples, containment relations have a strong relation to another important notion of algorithmic and structural graph theory: graph classes. Indeed, it is a well-known fact that a class of graph is closed under a containment relation if and only if this class of graph admits a (not necessarily finite) set of graphs that are forbidden with respect to this containment relation. Because of this bond between containment relations and graph classes, algorithms that quickly decide if a graph contains another can in turn be used to devise fast algorithms that recognize those graph classes that have

a finite set of forbidden graphs with respect to the containment relation under consideration. This tight bond between containment relations and graph classes obviously also holds from a purely combinatorial perspective.

Yet another important family of problems in algorithmic graph theory that also shares a tight bond, albeit less direct, with containment relation problems is, as we mentioned above, graph modification problems. The problems in this family typically ask if it is possible to perform a specified (generally small) number of elementary operations so that the resulting graph satisfies some property. It is interesting to observe that some problems are at the same time containment relation and graph modification problems. This is for instance the case for the four problems mentioned previously, i.e., finding a clique, an independent set, a Hamiltonian path or a Hamiltonian cycle. The first two of these problems can be reformulated to fit within the scope of graph modification problems by asking if it is possible to delete a given number of vertices so that the remaining vertices are all pairwise adjacent (respectively non-adjacent). Similarly, finding a Hamiltonian path (respectively cycle) can be seen as finding a set of edges whose deletion yields a path (respectively cycle).

1.2 Overview of Part I

This thesis consists of two main parts. In this current Part I, we give the necessary background and an overview of previously known results. Part II consists of new results that form the scientific contribution of this thesis.

We conclude this first chapter by briefly describing the contents of the next chapters that form the first part of this thesis. Chapter 2 contains all the definitions, notation and terminology that will be used in the subsequent chapters of Part I of this thesis. Various notions from graph theory, graph classes, computational complexity are defined, as well as a specific type of logic called monadic second order logic. Chapters 3 and 4 form the main part of Part I. The goal of Chapters 3 and 4 is to provide a broad overview of the area of containment relations in graphs. In particular, all results mentioned in these chapters are previously known.

Chapter 3 deals with algorithmic aspects of containment relations in graphs. We first define several operations on graphs that allow us to describe various well-known containment relations, and subsequently provide well-known alternative definitions of these containment relations and discuss how they relate to each other. We then dive into the main topic of

this chapter and survey results concerning the complexity of deciding containment relations on general graphs. We provide examples of algorithms that exhibit the tight relation between some of the containment relations discussed previously and the existence of vertex-disjoint paths. We also show how to express most of the containment relations considered in this thesis in monadic second order logic, which in turn has applications to design efficient algorithms for these problems on special classes of graphs. Following this idea, we conclude Chapter 3 with a list of results regarding the complexity of the decision problems associated with various containment relations, when restrictions are imposed on the structure of either the source and/or the target graph.

Chapter 4 deals with the other main aspect of containment relations on graphs, namely combinatorial aspects. As we mentioned, one of the main questions regarding containment relations in structural graph theory is to describe graphs that exclude a fixed target graph, which we will discuss first in Chapter 4. Another important point that we will discuss is the notion of well-quasi-ordering. We will conclude the chapter with a discussion on the notion of obstructions, which we already mentioned as the set of graphs that must be forbidden with respect to a certain containment relation in order to obtain a certain graph class.

Chapter 5 concludes Part I with a few examples of containment relations not discussed in Chapters 3 and 4, a list of open problems in the area of containment relations, and a short summary of the papers that form the scientific contribution of this thesis, i.e., Part II.

Chapter 2

Notation and terminology

This section contains most of the definitions and notation that will be required throughout Part I. Additional notation and definitions will be given when they are needed.

2.1 Graph theory

We start by giving basic graph theoretic definitions. We refer to the monograph by Diestel [40] for an in-depth introduction to graph theory. A graph G is an ordered pair of sets (V, E) with $E \subseteq V \times V$, where V is the set of *vertices* of G, while E is the set of *edges*. We will also use $V(G)$ and $E(G)$ to denote the vertex and edge set of a graph respectively. If the graph G is a *multigraph*, then $E(G)$ is a multiset, i.e., a set with repeated elements. We say that G is a *simple graph* when we want to emphasize that we do not allow G to be a multigraph. The number of occurences of an element e in the edge set of a multigraph G is referred to as the *multiplicity* of e, and is denoted by $\text{mult}_G(e)$.

An *isomorphism* between two graphs G and H is a bijection $\phi : V(G) \to V(H)$ such that for every pair of vertices $u, v \in V(G)$ we have $\text{mult}_G(uv) = \text{mult}_H(\phi(u)\phi(v))$. Note that if G and H are simple graphs, this is equivalent to having $uv \in E(G)$ if and only if $\phi(u)\phi(v) \in E(H)$, for every pair of vertices $u, v \in V(G)$. If there is an isomorphism from G to H, then we say that G and H are *isomorphic*, and write $G = H$. For any set S of vertices of G, we denote by $G - S$ the graph $G' = (V \setminus S,\ E \setminus (S \times S))$. Given a set $X \subseteq V(G)$, the *subgraph of G induced by X*, denoted $G[X]$, is the graph with vertex set X and edge set $\{uv \in E(G) \mid u, v \in X\}$.

Two distinct vertices $u, v \in V(G)$ such that $uv \in E(G)$ are said to be *adjacent* in G, and u and v are called the *endpoints* of the edge uv. If two

vertices u, v are non-adjacent in G, then $\text{mult}_G(uv) = 0$. Moreover, u and v are said to be *incident* with uv, and two edges $uv, xy \in E(G)$ are said to be incident if they share an endpoint. An edge whose endpoints are both vertex u is called a *loop at* u. Simple graphs are exactly multigraphs without loops in which every edge has multiplicity at most 1. The set of vertices adjacent to a vertex u is called the set of *neighbors* of u in G and is denoted by $N_G(u)$, while the set $N_G(u) \cup \{u\}$ is called the *closed neighborhood* of u and is denoted by $N_G[u]$. The number of neighbors of u in G is called the *degree* of u in G and is denoted $d_G(u)$. We omit the subscripts when there is no ambiguity. For a set S of vertices, we have $N(S) = \bigcup_{u \in S} N(u) \setminus S$ and $N[S] = N(S) \cup S$. A vertex u such that $N[u] = V(G)$ is called a *universal vertex* of G.

A *path* in G is a sequence of vertices $x_1, x_2, \ldots, x_n$ such that $x_i x_{i+1} \in E(G)$ for every $1 \leq i \leq n-1$. If we have $x_1 = x_n$, then we obtain a *cycle*. The number of edges of a cycle or a path is called its *length*. A path $x_1, \ldots, x_n$ is *chordless*, or equivalently *induced*, if $x_i x_j \notin E(G)$ whenever $|i - j| \neq 1$; a cycle is chordless if it can be obtained from a chordless path on the same number of vertices by adding the edge $x_1 x_n$. A chordless cycle of length at least 4 is called a *hole*. A graph isomorphic to a chordless path on n vertices is denoted by P_n, and a graph isomorphic to a chordless cycle on n vertices is denoted by C_n. The *complete graph* of size n consists of n pairwise adjacent vertices and denoted by K_n. A set of pairwise adjacent vertices is called a *clique*, while a set of pairwise non-adjacent vertices is called an *independent set*. The *complement* of a simple graph G, denoted $\overline{G}$, is the graph with vertex set V and edge set $\{uv \mid u, v \in V(G) \wedge uv \notin E(G)\}$. The graph obtained from $\overline{K_n}$ by adding a universal vertex is called a *star* and is denoted by $K_{1,n}$, and the star $K_{1,3}$ is called the *claw*. The graph obtained from $K_{1,3}$ by replacing its three edges by paths of length i, j, k is denoted by $S_{i,j,k}$. The graph obtained from C_n by adding a universal vertex is called a *wheel* and is denoted by W_n.

A graph G is *connected* if there is a path between every pair of vertices of G. A *separator* S of G is a set of vertices such that G is connected but $G - S$ is not. A vertex u such that $\{u\}$ is a separator of G is called a *cut-vertex* of G. A set of vertices $S \subseteq V(G)$ is called a *biconnected component* of G if, for every set $S' \supset S$, $G[S']$ has a cut-vertex, and either $G[S]$ does not have a cut vertex, or $G[S]$ is isomorphic to K_2. A partition (A, B) of the vertices of G is called a *cut* (sometimes edge-cut). An edge uv such that there is a cut (A, B) with $u \in A, v \in B$ and $N(A) = b$ is called a *bridge*.

A *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{X}$ is a collection of subsets of $V(G)$, called *bags*, and $\mathcal{T}$ is a tree whose vertices, called *nodes*, are the bags of $\mathcal{X}$, such that the following three properties are satisfied.

- $\bigcup_{X \in \mathcal{X}} X = V(G)$,
- for each edge $uv \in E$, there is a bag $X \in \mathcal{X}$ with $u, v \in X$,
- for each $x \in V(G)$, the set of bags containing x forms a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{T}, \mathcal{X})$ is the size of a largest bag in $\mathcal{X}$ minus 1. The *treewidth* of G is the minimum width over all possible tree decompositions of G. In [17], Bodlaender gave an algorithm that decides if G has treewidth at most k and runs in linear time, for every graph G and fixed integer k. A tree decomposition where $\mathcal{T}$ is a path is called a *path decomposition*. A graph has *pathwidth* at most k if it has a path decomposition of width k.

Some other width parameters will be briefly mentioned later in this thesis. These parameters are called *branch-width*, *clique-width*, *rank-width* and *carving-width*. We do not provide definitions for branch-width, clique-width and rank-width, as these will not be required for the scientific results of this thesis. At this point, it is sufficient to note that, as shown by Courcelle and Rotics [32], for any graph G, the clique-width of G is at most $3 \cdot 2^{k-1}$, where k is the treewidth of G. Equivalently, every class of graphs having bounded treewidth also has bounded clique-width. Please refer to the survey by Hlinený, Oum, Seese and Gottlob [85] for an introduction to branch-width, clique-width and rank-width. Carving-width is defined in Chapter 10.

Let $\leq$ be a binary relation on a set P. The pair $(P, \leq)$ is a *partially ordered set* if for every $x, y, z \in P$, we have:

- $x \leq x$, i.e., $\leq$ is reflexive;
- $x \leq y$ and $y \leq z \Rightarrow x \leq z$, i.e., $\leq$ is transitive; and
- if $x \leq y$ and $y \leq x$, then $x = y$, i.e., $\leq$ is antisymmetric.

We also say that $\leq$ is a *partial order* over P. If $x \leq y$ and $x \neq y$, then we naturally write $x < y$.

Let $\mathcal{G}$ be a graph class and $\leq$ a partial order over the set of all graphs. We say that $\mathcal{G}$ is *closed* under $\leq$ if for every graph $G \in \mathcal{G}$, every graph

$H \leq G$ also belongs to $\mathcal{G}$. We say that a graph H is an *obstruction* for $\mathcal{G}$ with respect to the relation $\leq$ if $H \notin \mathcal{G}$ and H is a minimal, i.e., for every $H' < H$, we have $H' \in \mathcal{G}$. We will generally refer to H as a $\leq$ *obstruction* for $\mathcal{G}$ and call the set $\mathcal{O}$ of all $\leq$ obstructions for $\mathcal{G}$ the $\leq$ *obstruction set* for $\mathcal{G}$.

2.2 Graph classes

In this section we provide definitions for several classes of graphs that will be used later on. A graph class $\mathcal{G}$ is a (generally infinite) set of graphs. For two classes of graphs $\mathcal{G}$ and $\mathcal{G}'$, we say that $\mathcal{G}$ is a *subclass* of $\mathcal{G}'$ if every graph that belongs to $\mathcal{G}$ also belongs to $\mathcal{G}'$. In such a case, we write $\mathcal{G} \subseteq \mathcal{G}'$. This naturally defines a hierarchy of graph classes. The hierarchy of the classes described in this section can be found in Figure 2.1. For a comprehensive study of graph classes, please refer to the monographs by Brandstädt, Le and Spinrad [22], and by Golumbic [78], or to the online compendium of graph classes ISGCI [38].

Before we define the various graph classes, we need a few additional definitions. The *join* of two graphs G_1, G_2 with $V(G_1) \cap V(G_2) = \emptyset$ is the graph $G = (V(G_1) \cup V(G_2),\ E(G_1) \cup E(G_2) \cup (V(G_1) \times V(G_2))$. Similarly, the *disjoint union* of G_1 and G_2, denoted by $G_1 + G_2$, is the graph $G' = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. For any graph G and integer k, we denote by kG the graph obtained by taking the disjoint union of k copies of G. Let $\mathcal{F}$ be a family of nonempty sets. For a graph G, we say $\mathcal{F}$ is an *intersection model* of G if there exists a bijection ϕ from $\mathcal{F}$ to $V(G)$ such that two vertices $u, v \in V(G)$ are adjacent if and only if $\phi(u)$ and $\phi(v)$ intersect. In such a case, we also say that G is an *intersection graph* of $\mathcal{F}$. A graph is an intersection graph if it is an intersection graph for some family of sets. The sets in the intersection model often consist of geometric objects such as lines, circles or polygons.

Trees and forests: A *forest* is a graph without cycles. A tree is a connected *forest*.

Planar graphs and bounded genus graphs: A *planar graph* is a graph that can be embedded in the plane without edges crossing. A graph has genus at most g if it can be embedded in a surface of Euler genus g without crossing edges. Planar graphs are exactly graphs of genus 0. Let G be a planar graph embedded in the plane without edges crossing. A maximal region of the plane that does not contain vertices or edges of G

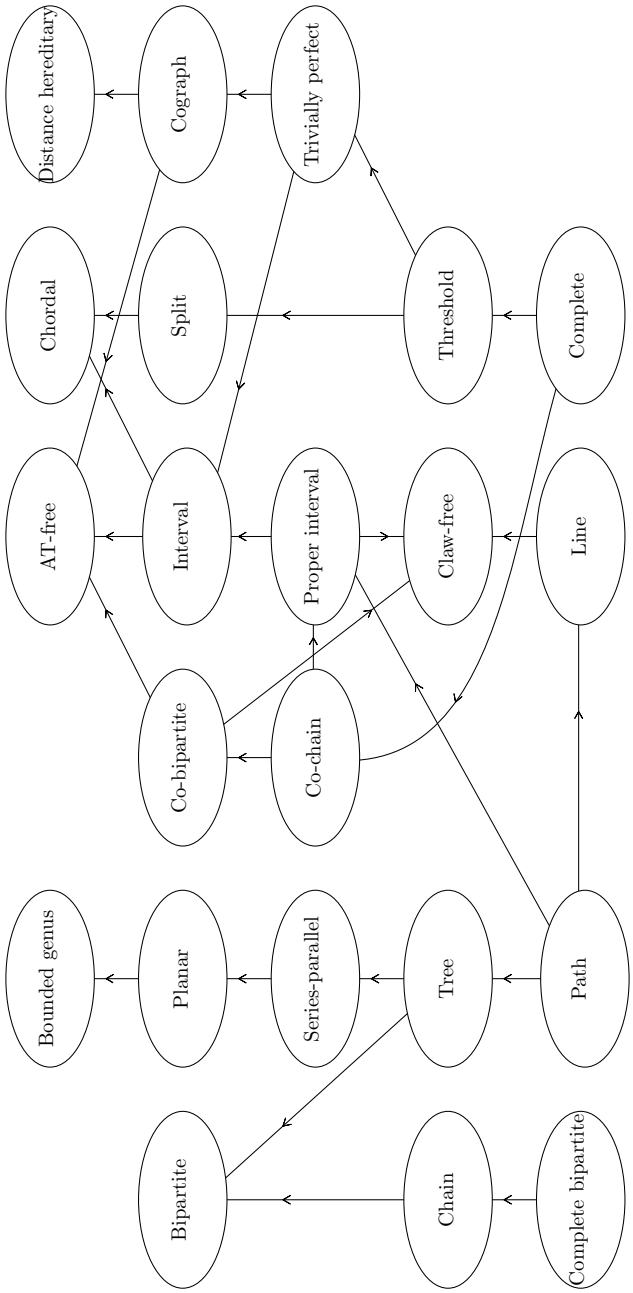


Figure 2.1: Hierarchy of the graph classes defined in this section. An arrow from class $\mathcal{G}$ to class $\mathcal{G}'$ means that $\mathcal{G} \subseteq \mathcal{G}'$.

is called a *face* of G. Every face f is bounded by a set of edges, and these edges are said to be incident with f. Let F denote the set of faces of G. The *dual* of G is the graph G^* with vertex set F and for every edge e of G, we add an edge between the two faces f, f' incident with e, or we add a loop at f if e is incident with only one face f.

Bipartite graphs, co-bipartite graphs, chain graphs, and complete bipartite graphs: A *bipartite graph* is a graph whose vertex set can be partitioned into two independent sets. A graph G is co-bipartite if $\overline{G}$ is bipartite. A graph G is a chain graph if it is bipartite, with partition, say, (X, Y), and for every two vertices $u, v \in X, N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$ holds. This property then automatically holds for all vertices of Y as well. We denote by $K_{n,m}$ the bipartite graph having partition (X, Y) such that $|X| = n, |Y| = m$ and $E(G) = \{xy \mid x \in X, y \in Y\}$. A graph that is isomorphic to $K_{n,m}$ for some value of n and m is called *complete bipartite*.

Bounded degree graphs: For any integer d, a graph has degree at most d if every vertex in it has degree at most d. We refer to graphs of maximum degree at most 3 as *subcubic graphs*.

Chordal graphs: A graph is *chordal* if and only if it does not contain an induced (chordless) cycle of length at least 4 as an induced subgraph. Chordal graphs are also well-known to be the intersection graphs of subtrees of a tree.

Split graphs and threshold graphs: A *split* graph is a graph whose vertex set can be partitioned into a clique and an independent set. Split graphs are well-known to form a proper subclass of chordal graphs. A graph G is a *threshold graph* if it is a split graph and for every two vertices $u, v \in V(G), N[u] \subseteq N[v]$ or $N[v] \subseteq N[u]$ holds. Alternatively, a graph is a threshold graph if it can constructed from the empty graph by repeatedly adding either an isolated vertex or a dominating vertex.

AT-free graphs: Three vertices of a graph form an *asteroidal triple* (AT) if every two of them are connected by a path avoiding the neighborhood of the third. A graph is *AT-free* if it does not contain any asteroidal triple.

Interval graphs and proper interval graphs: An *interval graph* is the intersection graph of a family of intervals of the real line. A graph is an interval graph if and only if it is chordal and AT-free [104]. A *proper interval graph* is an interval graph that has an intersection model in which no interval properly contains another. A graph is a proper interval graph if and only if it is a unit interval graph, i.e., it has an intersection model in which every interval has unit length [124].

Cographs: A graph is a *cograph* if it can be constructed from single vertices by disjoint union and join operations. Cographs are well-known to be exactly those graphs that do not contain P_4 as an induced subgraph (see e.g., [22, 78]).

Trivially perfect graphs: A graph is a *trivially perfect graph* if for every induced subgraph H, $\alpha(H) = |C(H)|$, where $\alpha(H)$ is the size of the largest independent set of H and $C(H)$ is the set of all its maximal cliques. A graph is trivially perfect if and only if it does not contain P_4 or C_4 as an induced subgraph [77, 150, 151]. An interesting characterization of trivially perfect graphs, which will be central in Chapter 6, is that trivially perfect graphs are exactly the graphs obtained from a rooted tree by adding an edge between two vertices whenever one is a descendant of the other [150, 151].

Line graphs: The *line graph* $L(G)$ of G is the graph with vertex set $E(G)$ in which two vertices $x, y \in E(G)$ are adjacent in $L(G)$ if and only if the edges x and y are incident in G. A graph is a line graph if it is the line graph of some graph.

Distance hereditary graphs: A graph G is *distance-hereditary* if, for every set S such that $G[S]$ is connected and for every $x, y \in S$, the length of a shortest path between x and y in G is equal to the length of a shortest path between x and y in $G[S]$.

Series-parallel graphs: The S operation consists in replacing an edge uv by two edges uw, wv in series, where w is a new vertex made adjacent only to u and v. The P operation consists in increasing the multiplicity of an edge by 1. Let G be a multigraph. G is *series-parallel* if its maximal 2-connected components can be generated from a loop by applying a sequence of S and P operations.

$\mathcal{H}$-free graphs: A graph G is *$\mathcal{H}$-free*, for some family of graphs $\mathcal{H}$, if there is not set X of vertices of G and graph $H \in \mathcal{H}$ such that $G[X] = H$. If $\mathcal{H}$ contains a single graph H, then we simply say that G is H-free. A graph is *claw-free* if it is $K_{1,3}$-free. Cographs are exactly P_4-free graphs.

2.3 Parameterized complexity

In this section we define the three basic families of running time of algorithms that we will use in the next chapters, together with their associated concept of hardness. For more detailed information on parameterized complexity and parameterized algorithms, see the monographs by Downey and Fellows [48], Flum and Grohe [59], and Niedermeier [117]. Note that in this thesis we only consider decision problems.

The following definitions are adapted from the monograph by Flum and Grohe [59]. A *problem* is a language over a finite alphabet Σ. We refer to sets $Q \subseteq \Sigma^*$ as a *non-parameterized problem*, or simply a *problem* when there is no confusion. A non-parameterized problem Q is represented in the form

Problem Name
Instance: $x \in \Sigma^*$.
Question: Is x in Q?

As an example, consider the problem Independent Set.

Independent Set
Instance: A graph G and an integer k.
Question: Does G have an independent set of size at most k?

We now define the notions of parameterization and parameterized problem.

Definition 2.1. *Let Σ be a finite alphabet.*

- *A* parameterization *of Σ^* is a mapping $\kappa : \Sigma^* \to \mathbb{N}$ that is polynomial time computable.*
- *A* parameterized problem *(over Σ) is a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ of strings over Σ and a parameterization κ of Σ^**

A parameterized problem (Q, κ) is represented in the form

κ-Problem Name
Instance: $x \in \Sigma^*$.
Parameter: $\kappa(x)$.
Question: Is x in Q?

Following with the example of Independent Set, consider the same problem, parameterized by the solution size.

k-Independent Set
Instance: A graph G.
Parameter: k.
Question: Does G have an independent set of size at most k?

In the particular case of graph containment relation problems, which form the main focus of this thesis, the input usually consists of a source graph G and a target graph H, and $|V(H)|$ is most typically used as parameter. Hence, the non-parameterized and parameterized forms of the problem are denoted by Problem Name and H-Problem Name respectively. Note that we slightly abuse notion in the second case, as we should be writing $|V(H)|$-Problem Name instead.

Finally, we discuss tractability of problems in terms of both classical and parameterized complexity.

Definition 2.2. *Let Σ be an alphabet and $\kappa : \Sigma^* \to \mathbb{N}$ a parameterization.*

(i) A problem Q is polynomial-time solvable *if there is an algorithm that decides Q in time $O(|x|^c)$ for some constant c.*

(ii) A parameterized problem (Q, κ) is fixed-parameter tractable *(or just FPT for short) with respect to κ if there is an algorithm that decides Q in time $f(\kappa(x)) \cdot |x|^c$ for some constant c and some computable function f,*

(iii) A parameterized problem (Q, κ) is polynomial-time solvable for every fixed value of κ *if there is an algorithm that decides Q in time $O(f(\kappa(x)) \cdot |x|^{\kappa(x)})$.*

The complexity class P contains all the problems that are polynomial-time solvable. Similarly, the class FPT contains all the fixed-parameter tractable parameterized problems, and the class XP contains all parameterized problems that can be solved in polynomial time for every fixed value of their parameter κ.

Additionally, an algorithm that runs in time $O(|x|^c)$, for some constant c, is referred to as a polynomial-time algorithm. An algorithm that runs in time $f(\kappa(x)) \cdot |x|^c$, for some constant c and some computable function f, is called an FPT algorithm. An algorithm that runs in time $O(f(\kappa(x) \cdot |x|^{\kappa(x)})$ is called a polynomial-time algorithm for every fixed value of κ, or an XP algorithm by abusing terminology.

Note in particular that if a problem is solvable in polynomial-time in its non-parameterized form, then it is FPT with respect to any parameter κ. Moreover, since we have $f(\kappa(x)) \cdot |x|^c \in O(x^{f(\kappa(x))})$ for any parameter κ, it follows that if a problem is FPT, then it is also solvable in polynomial time for every fixed value of its parameter κ. Hence, we may immediately conclude that $\mathsf{P} \subseteq \mathsf{FPT} \subseteq \mathsf{XP}$. However, the converse of these two statements is not necessarily true under standard complexity theoretic assumptions. Under the well-known assumption that $\mathsf{P} \neq \mathsf{NP}$, NP-hard problems do not admit polynomial-time algorithms, and parameterized problems that are NP-hard for some fixed value of their parameter κ do not admit XP algorithms. A class of hard problems also exists for the class FPT, namely the class W[1], which is the class of problems that admit a so-called *fixed-parameter tractable reduction* to the problem k-Clique, where k is the solution size. Under the assumption that $\mathsf{FPT} \neq \mathsf{W}[1]$, W[1]-hard problems do not admit FPT algorithms. In fact, W[1] is only the first class in the so-called W-hierarchy, which is composed of the classes

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[P] \subseteq \mathsf{XP}$$

In particular, proving that a problem is W[i]-hard for any $i \geq 1$ immediately implies that the problem is W[1]-hard, and therefore gives strong evidence that the problem does not admit an FPT algorithm. Within this thesis, we will not consider the levels of the hierarchy above W[1], as we work under the assumption that $\mathsf{FPT} \neq \mathsf{W}[1]$, and therefore proving W[1]-hard is sufficient to show that a problem is not likely to be in FPT.

The usual way of proving that a problem Q (or (Q, κ) in the parameterized case) is NP-complete is by providing a polynomial-time reduction from a known NP-complete problem to Q; to prove W[1]-hardness, we provide a fixed-parameter tractable reduction from a known W[1]-hard parameterized problem (Q', κ') to (Q, κ). For more details about the notion of NP-completeness and polynomial-time reductions, please refer e.g., to the monograph by Sipser [142]; for more details regarding the W-hierarchy and FPT reductions, please refer e.g., to the monograph by

Flum and Grohe [59]. Here we provide a few examples of classical problems known to be NP-complete (with and without parameterization) and W[1]-hard.

- Non-parameterized NP-complete problems: Satisfiability, Independent Set, Hamiltonian Path, etc.
- W[1]-hard problems: k-Independent Set, k-Dominating Set, etc.
- Parameterized NP-complete problems: 3-Satisfiability, and more generally k-Satisfiability for any fixed $k \geq 3$, 3-Coloring, etc.

A *kernel* of a parameterized problem (Q, κ) over the alphabet Σ is a polynomial time computable function $K : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$ we have $x \in Q$ if and only if $K(x) \in Q$, $\kappa(K(x)) \leq \kappa(x)$, and $|K(x)| \leq h(\kappa(x))$ for some computable function h. It is a well-known fact that a parameterized problem (Q, κ) is FPT if and only if it has a kernel. However, the size of the kernel is very often an exponential function of the parameter. In some cases, we can obtain a kernel whose size is polynomial in the parameter, which is a desirable property. More formally:

Definition 2.3. *A kernel K of a parameterized problem (Q, κ) over the alphabet Σ is* polynomial *if there is a constant $c \in \mathbb{N}$ such that $|K(x)| \in O(\kappa(x)^c)$ for every $x \in \Sigma^*$.*

If the parameterized problem (Q, κ) takes as input a graph G, then a polynomial *vertex-kernel* is a kernel K of (Q, κ) such that for every input graph G, $|V(K(G))| \in O(\kappa(G)^c)$, for some constant c. For example, a linear vertex-kernel will output a graph on $O(\kappa(G))$ vertices.

2.4 Monadic Second Order Logic

We end this chapter by giving the definition of a standard logic which will be used later in this thesis.

A Monadic Second Order Logic sentence is a logical formula using the following operations:

- logical connectives $\wedge, \vee, \rightarrow, \neg, =, \neq$,
- quantifiers $\forall, \exists$ over vertex/edge variables,
- predicate $adj(u, v)$: vertices u and v are adjacent,

- predicate $inc(e, v)$: edge e is incident to vertex v,
- quantifiers $\forall, \exists$,
- $\in, \subseteq$ for vertex/edge sets.

Monadic second order (MSO) logic is a powerful fragment of second order logic that can represent many natural graph properties, while having attractive algorithmic properties, as we will see several examples of in this thesis.

We conclude this section with an example by giving the MSO-sentence that expresses that a graph is 3-colorable:

$$\begin{aligned}\phi_{3-col}(G) =& \exists X, Y, Z \subseteq V(G) : \\ & (\forall u \in V(G)(u \in X \vee u \in Y \vee u \in Z)) \wedge \\ & (\neg \exists u \in V(G) : \\ & \quad ((u \in X \wedge u \in Y) \vee (u \in X \wedge u \in Z) \vee (u \in Y \wedge u \in Z))) \wedge \\ & (\forall u, v \in V(G) \\ & \quad ((u, v \in X) \vee (u, v \in Y) \vee (u, v \in Z)) \rightarrow \neg adj(u, v)))\end{aligned}$$

Chapter 3

Algorithmic aspects of containment relations in graphs

As we mentioned in Chapter 1, the problems associated with graph containment relations typically encompass many other well-studied problems and are often used as a basis to provide powerful metatheorems, such as the one obtained by Robertson and Seymour by combining the results from [129] and [131] (which will be discussed in details later).

3.1 Graph operations and containment relations

In this section we formally define several standard operations on graphs and the containment relations that can be obtained by combining them. Before defining specific containment relations, we first suggest the following as a formal definition for the notion of containment relation.

Definition 3.1 (Containment relation)**.** *A binary relation $\leq$ over the set of graphs is a containment relation if it is a partial order, i.e., it is reflexive, transitive and antisymmetric.*

The choice to use such a definition for containment relations might seem obvious at first glance. Throughout the literature, it is rather common to come across such terminology as "minor order", for instance, and the term "partial order" itself appears frequently to refer to such kind of relations. However it is not entirely obvious that all notions that would naturally fall under the label of "containment relations" are indeed partial orders. Examples include the vertex minor relation, which will be

discussed in Chapter 5, since two non-isomorphic graphs can be vertex minors of each other.

A common way of defining containment relations in graphs is by using operations, i.e., functions that take a graph as input and produce a graph as output. We now define the operations that are used in this thesis; note that these operations are defined for both simple graphs and multigraphs, with and without self-loops. Let $G = (V, E)$ be a graph.

Vertex deletion (VD): The operation of *deleting a vertex* u of G, denoted $G - u$, yields a graph G' with vertex set $V \setminus \{u\}$ and edge set $E \setminus \{uy \mid y \in V\}$.

Edge deletion (ED): The operation of *deleting an edge* e of G, denoted $G - e$, is defined similarly, yielding graph $G' = (V, E \setminus \{e\})$.

Edge contraction (EC): *Contracting* an edge $e = uv$ of G, with $u \neq v$, denoted G/e, yields graph G' with vertex set $(V \setminus \{u, v\}) \cup \{w\}$, such that $w \notin V$, and edge set $(E \setminus \{xy \in E \mid x \in \{u, v\}, y \in V(G)\}) \cup \{wz \mid z \in N(u) \cup N(v)\}$.

Lift (L): The operation of *lifting* two edges $\{uv, vw\}$ of G, where uv and vw are incident, denoted $G \vee \{uv, vw\}$, yields graph $G' = (V, (E \setminus \{uv, vw\}) \cup \{uw\})$. See Figures 3.1, 3.2 and 3.3 below for an illustration.

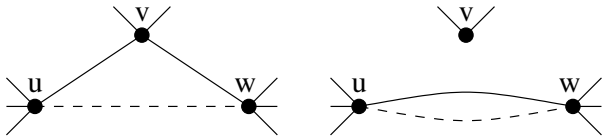


Figure 3.1: Lift $\{uv, vw\}$ when u, v and w are pairwise distinct.

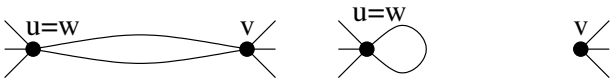


Figure 3.2: Lift $\{uv, vw\}$ when $u = w$ and v is distinct from u.

Vertex dissolution (VDi): The operation of *dissolving* a vertex v of G, where $d_G(v) = 2$, denoted $G \wedge \{v\}$, yields graph G' with vertex set $V \setminus \{v\}$ and edge set $(E \setminus \{uv, vw\}) \cup \{uw\}$. It is also interesting to note

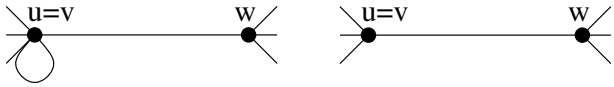


Figure 3.3: Lift $\{uv, vw\}$ when $u = v$ and w is distinct from u.

that $G' = (G \vee \{uv, vw\}) - v = G/uv = G/vw$.

The first three of these operations, namely vertex and edge deletion and edge contraction, are standard and widely used in every area of graph theory. The two last operations, lift and vertex dissolution, are maybe less well-known. The notion of lift was introduced by Lovász [109, 110] under the name *splitting-off* as a way to maintain the edge-connectivity of a graph. See e.g., [102, 153] for a list of applications, which include edge-connectivity augmentation problems and network design problems, among others. The last operation, vertex dissolution, can be seen as a special case of lift and vertex deletion, where the vertex incident with the two edges lifted is immediately deleted. Vertex dissolution is also a special case of contraction, where we impose one of the two endpoints of the contracted edge to have degree exactly 2. An immediate consequence of these two observations is that, when we define a containment relation, if we allow the the contraction operation or lift together with vertex deletion, then we also allow dissolution. Another point worth mentioning is that edge deletion and vertex dissolution both have a "dual" operation, respectively edge addition and edge subdivision. Edge addition is defined naturally, and subdividing edge uv consists in deleting uv and adding a new vertex w and edges uw and vw.

Given the set $\mathcal{F} = \{\text{VD,ED,EC,VDi,L}\}$ of operations described above, we can define various containment relations by considering the power set of $\mathcal{F}$. Every possible subset $\mathcal{F}'$ of $\mathcal{F}$ yields a well-defined containment relation where we are only allowed to use the operations that appear in $\mathcal{F}'$. In other words, the set of operations $\mathcal{F}'$ defines a relation $\leq$ such that for any two graphs G, H we have $H \leq G$ whenever a graph isomorphic to H can be obtained from G by sequentially applying operations in $\mathcal{F}'$. These relations are presented in Table 3.1, for example the "minor" relation allows to delete vertices and edges, contract edges and dissolve vertices, but the lift operation is not allowed.

However, one may observe that we have 5 operations in $\mathcal{F}$, and hence we should have $2^5 = 32$ different containment relations, while only 14 are presented in Table 3.1. The remaining 18 relations consist of:

Containment Relation	VD	ED	EC	VDi	L
Minor	yes	yes	yes	yes	no
Induced minor	yes	no	yes	yes	no
Topological minor	yes	yes	no	yes	no
Induced topological minor	yes	no	no	yes	no
Immersion	yes	yes	no	yes	yes
Induced immersion	yes	no	no	yes	yes
Lift-contraction	no	no	yes	yes	yes
Lift-minor	no	yes	yes	yes	yes
Contraction	no	no	yes	yes	no
Dissolution	no	no	no	yes	no
Subgraph	yes	yes	no	no	no
Induced subgraph	yes	no	no	no	no
Spanning subgraph	no	yes	no	no	no
Isomorphism	no	no	no	no	no

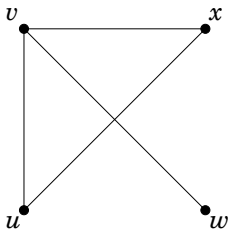
Table 3.1: Fourteen known containment relations in terms of the graph operations vertex deletion, edge deletion, edge contraction, vertex dissolution and lift.

- 6 relations where vertex dissolution is not allowed, but contraction or lift and vertex deletion are: we do not consider these relations for the reasons mentioned above;
- 6 relations other than spanning subgraph and lift-minor that allow edge deletion but not vertex deletion;
- 3 relations that allow only lift or edge contraction;
- 2 relations that allow vertex deletion, edge contraction, vertex dissolution and lift;
- 1 relation that allows only lift and vertex dissolution.

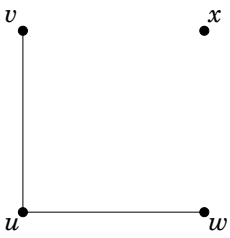
To the best of our knowledge, none of these relations has been studied yet, except for the lift relation that was studied in the context of splitting-off (see e.g [110, 153]).

Before we move on to the next section, we would like to point out that for a set of operations $\mathcal{F}' \subseteq \mathcal{F}$, the order in which the operations are applied may or may not matter. For example, it can be observed

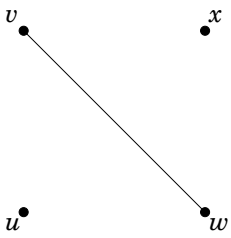
that vertex and edge deletions can be applied in any possible order and will always yield the same graph, up to isomorphism. However, for some sets of operations, the order of the sequence may actually matter. For an example, consider the lift operation defined for simple graphs, i.e., if a lift creates multiple copies of an edge, we reduce its multiplicity to one and keep the graph simple. Observe that applying lifts $\{uv, vw\}$ and $\{ux, xv\}$ to the graph shown in Figure 3.4 does not yield the same graph if the order in which lifts are applied changes. Indeed, the graphs $(G \vee \{uv, vw\}) \vee (\{ux, xv\}$ and $(G \vee \{ux, xv\}) \vee (\{uv, vw\}$ are easily seen to be non-isomorphic.



(a) G



(b) $(G \vee \{uv, vw\}) \vee (\{ux, xv\}$



(c) $(G \vee \{ux, xv\}) \vee (\{uv, vw\}$

Figure 3.4: Applying lifts $\{uv, vw\}$ and $\{ux, xv\}$ in a different order yields two non-isomorphic graphs.

3.2 Alternative definitions of some relations

In the previous section, we defined several operations on graphs and the various containment relations obtained by combining these operations. In several cases, the definition of a containment relation in terms of operations is sufficient to allow us to derive efficient algorithms to decide this relation. This is the case for example for the subgraph and induced subgraph relations, where the pattern is contained "locally", i.e., if a graph G contains a graph H, then H can be found in a relatively small part of G,

comparable to $|V(H)|$. Therefore, if the pattern graph is itself sufficiently small, then it is possible to decide if G contains H by trying every "small part" of G and checking if H can be found in it. However, some operations do not have this local behavior. An example of such operation is the edge contraction, since by successively contracting edges in a graph G, parts of the graph that were originally lying far from each other can be brought together in order to form a new pattern. In this sense, the contraction operation is more related to vertex partitioning problems, such as e.g., Coloring, than to vertex subset problems, such as e.g., Clique.

Another issue with defining containment relations via sets of operations is that it constructs the pattern graph H dynamically, i.e., by applying the operations in the sequence one after the other. However, it is often easier to work with more "static" structures. We now provide well-known alternative definitions of several containment relations. The purpose of these definitions is twofold: first, they provide the "static" structures mentioned above and are the basis for designing efficient algorithms for deciding these relations, and second, these definitions illustrate some of the differences between some relations that would otherwise appear very similar, e.g., the minor and topological minor relations.

Consider for instance the following well-known alternative definition of minor, induced minor and contraction.

Definition 3.2 (Witness structure)**.** *Let G and H be two graphs such that $V(H) = \{v_1, \ldots, v_{|V(H)|}\}$. An H-witness structure of G is a family $\mathcal{W} = \{W_1, \ldots, W_{|V(H)|}\}$ of non-empty subsets of $V(G)$ such that for every $1 \leq i \neq j \leq |V(H)|$, the following holds:*

- *$W_i \cap W_j = \emptyset$, and*
- *$G[W_i]$ is connected.*

In addition,

(i) $\mathcal{W}$ is an H-minor witness structure if, for every $v_i, v_j \in V(H)$, if $v_iv_j \in E(H)$, then there exist vertices $x \in W_i, y \in W_j$ such that $xy \in E(G)$;

(ii) $\mathcal{W}$ is an H-induced minor witness structure if, for every $v_i, v_j \in V(H)$, $v_iv_j \in E(H)$ if and only if there exist vertices $x \in W_i, y \in W_j$ such that $xy \in E(G)$;

(iii) $\mathcal{W}$ is an H-contraction witness structure if $\bigcup_{i=1}^{|V(H)|} W_i = V(G)$ and for every $v_i, v_j \in V(H)$, $v_iv_j \in E(H)$ if and only if there exist vertices $x \in W_i, y \in W_j$ such that $xy \in E(G)$.

For simplicity, we use the term *H-witness structure* when it is clear from the context which containment relation is being considered. Witness structures are a powerful tool when working with the minor, induced minor and contraction relations. In particular, they will be used extensively in Chapters 6 and 7. It remains to show that witness structures are indeed a characterization of their associated containment relations. It is well-known that a graph G contains a graph H as a minor, induced minor or contraction if and only G has an H-witness structure, where the type of witness structure depends on the containment relation being considered. This fact is stated in Propositions 3.3, 3.4 and 3.5. For completeness and broad background coverage, we provide a proof here. We provide a proof only for Proposition 3.3, as those of Propositions 3.4 and 3.5 are nearly identical.

Proposition 3.3. *A graph G contains a graph H as a minor if and only if G has an H-minor witness structure.*

Proof. First we show that if G has an H-minor witness structure $\mathcal{W} = \{W_1, \ldots, W_{|V(H)|}\}$, then G contains H as a minor. Observe that if, for every set $W \in \mathcal{W}$, we contract the vertices of $\mathcal{W}$ to a single vertex, and we delete every vertex $w \notin \bigcup_{i=1}^{|V(H)|} W_i$, then we obtain a graph G' that contains H as a spanning subgraph. Since G contains G' as a minor, it follows that G contains H as a minor as well.

For the reverse direction, we consider the sets V' and E' of deleted vertices and contracted edges respectively, and exhibit an H-witness structure $\mathcal{W}$ of G. Let $\phi : V(H) \to V((G - V')/E')$ be a spanning subgraph isomorphism from H to $(G - V')/E'$. We know that ϕ exists since H can be obtained from G by deleting V', contracting E' and deleting some edges in the resulting graph. Consider the graph $G' = (V(G) - V', E')$. Since G can be contracted to H, there is a bijection ϕ' from $V((G - V')/E')$ to the set of connected components of G', such that, for every two vertices u, v of $V((G - V')/E')$, u and v are adjacent in $(G - V')/E'$ if and only if there are vertices $x \in \phi'(u)$ and $y \in \phi'(v)$ such that $xy \in E(G')$. We can now describe an H-witness structure of G: for every vertex $u \in V(H)$, we have $W(u) = \phi'(\phi(u))$.

It only remains to show that $\mathcal{W} = \bigcup_{u \in V(H)} W(u)$ is indeed an H-witness structure of G. Observe first that the set $\phi(x)$ is connected in G' for every vertex $x \in V((G - V')/E')$, and hence $\phi(\phi'(u))$ is connected for every vertex $u \in V(H)$. Moreover, for every edge $uv \in E(H)$, we have $\phi(u)\phi(v) \in E((G - V')/E')$, and there exists $x \in \phi'(\phi(u)), y \in \phi'(\phi(v)$ such that $xy \in E(G')$. Since G' is a subgraph of G, we conclude that for

every edge $uv \in E(H)$, there exist adjacent vertices in $W(u)$ and $W(v)$ for every edge $uv \in E(H)$. Hence $\mathcal{W}$ is indeed an H-witness structure of G. □

Proposition 3.4. *A graph G contains a graph H as an induced minor if and only if G has an H-induced minor witness structure.*

Proposition 3.5. *A graph G contains a graph H as a contraction if and only if G has an H-contraction witness structure.*

For the topological minor and induced topological minor relations, it is also possible to give a definition in terms of existence of a connected set of vertices with some specific properties, as we just did for minor, induced minor and contraction. However, this definition does not appear very natural in the case of (induced) topological minor. Instead, the alternative definitions for the topological minor, induced topological minor and immersion relations rely on the existence of vertex-disjoint, mutually induced disjoint, and edge-disjoint paths respectively. As with Propositions 3.3, 3.4 and 3.5, we provide only a proof for Proposition 3.6, as the proofs for Propositions 3.7 and 3.8 are very similar.

Proposition 3.6. *A graph G contains a graph H as a topological minor if and only if there exists a set S of vertices of G and bijection $\phi : V(H) \to S$ such that:*

- *for every edge $uv \in E(H)$, there exists a path P_{uv} with endpoints $\phi(u), \phi(v)$,*
- *for every two distinct edges $e, e' \in E(H)$, the paths P_e and $P_{e'}$ are internally vertex-disjoint.*

Proof. We first prove that if the bijection ϕ and the paths P_{uv} exist, then G contains H as a topological minor. We prove this by giving a set of vertex deletions, edge deletions and vertex dissolution in G that yield a graph isomorphic to H. Starting from G, we delete every edge that does not belong to a path P_{uv} and every vertex of G that is not the image by ϕ of some vertex of H, and we dissolve every internal vertex of every path P_{uv}. This yields a graph G' whose vertex set is exactly those vertices that are the image by ϕ of the vertices of H. We now claim that ϕ is an isomorphism from H to G'. This follows from the fact that every edge of H corresponds to a path P_{uv} in G, all internal vertices of which have been dissolved, yielding a single edge. Therefore, for every pair of vertices $u, v \in V(H)$, we have $\text{mult}_H(uv) = \text{mult}_{G'}(\phi(u)\phi(v))$, which immediately

implies that G' and H are isomorphic, and hence G contains H as a topological minor.

We now prove the converse direction. Assume that G contains H as a topological minor. Let S and D be sets of vertices, and F a set of edges such that $G' = ((G - S) - F) \wedge D$ is isomorphic to H, with ϕ an isomorphism from H to G'. We prove that for every edge xy of G', there is a path P_{xy} in G such that x and y are the endpoints of P_{xy}, and the paths P_{xy} are pairwise internally vertex-disjoint. This is easily observed from the fact that in the graph $(G - S) - F$, all the vertices in D have degree 2, and therefore form paths between the vertices of $V(G) \setminus (S \cup D)$, and for every pair of vertices x, y of $V(G) \setminus (S \cup D)$, the number of paths with endpoints x, y and whose internal vertices all belong to D is exactly $\mathrm{mult}_H(\phi^{-1}(x)\phi^{-1}(y))$. Therefore we can associate each edge $e \in E(H)$ with such a path, and these paths are easily observed to be internally vertex-disjoint. This completes the proof of Proposition 3.6. □

Proposition 3.7. *A graph G contains a graph H as an induced topological minor if and only if there exists a set S of vertices of G and bijection $\phi : V(H) \to S$ such that:*

- *for every edge $uv \in E(H)$, there exists a path P_{uv} with endpoints $\phi(u), \phi(v)$,*
- *for every two distinct edges $e, e' \in E(H)$, the paths P_e and $P_{e'}$ are mutually internally induced, i.e., internal vertices of P_e have no neighbor in $P_{e'}$ and vice-versa.*

Proposition 3.8. *A graph G contains a graph H as an immersion if and only if there exists a set S of vertices of G and bijection $\phi : V(H) \to S$ such that:*

- *for every edge $uv \in E(H)$, there exists a path P_{uv} with endpoints $\phi(u), \phi(v)$,*
- *for every two distinct edges $e, e' \in E(H)$, the paths P_e and $P_{e'}$ are edge-disjoint.*

We would like to point out that these definitions in terms of existence of paths with specific properties are actually somewhat more common than the definitions in terms of operations; this is especially true in the case of immersion, which is most commonly defined in the way of Proposition 3.8. Note also that the vertices of the set S in Propositions 3.6, 3.7

and 3.8 are often called *branch vertices*.

We would like to conclude this section with some additional comments regarding immersion. A natural question to ask regarding the definition given in Proposition 3.8 is whether the paths P_e are allowed to have vertices of S as internal vertices. Whether this is allowed or not gives rise to two definitions of immersion, namely *weak* and *strong* immersion. In terms of operations, weak immersion allows lifts to be performed naturally, whereas strong immersion requires that for every lift $\{uv, vw\}$ performed, vertex v has to be deleted at some point (not necessarily immediately after performing $\{uv, vw\}$, in particular it is possible to perform more lifts using v before deleting it). Although weak immersion is more common than strong immersion, the latter is also frequently considered, see e.g., [132, 152].

3.3 Relations between relations

In this section, we compare the different relations presented in Section 3.1 (see Table 3.1). From the definitions of these relations in terms of set of operations, it is clear that some of these relations are in a sense more "powerful" than some others. More precisely, we observe the following.

Observation 3.9. *Let $\leq_1, \leq_2$ be two containment relations defined by allowing the sets of operations F and F' respectively. If $F \subseteq F'$, then for every two graphs G, H it holds that if $H \leq_1 G$ then $H \leq_2 G$.*

Observation 3.9 immediately implies a hierarchy of containment relations, as illustrated in Figure 3.5 below.

However, in some particular cases depending on the structure of G and H, some of these relations "collapse" and become equivalent. Two well-known examples are given in the next two propositions.

Proposition 3.10. *Let G and H be two graphs such that G is subcubic. G contains H as an immersion if and only if G contains H as a topological minor.*

Proof. If G contains H as a topological minor, then G clearly contains H as an immersion as well, hence we only have to prove the converse statement. Assume that G contains H as an immersion. By Proposition 3.8, there exists a set S of vertices of G and bijection $\phi : V(H) \to S$ such that:

- for every edge $uv \in E(H)$, there exists a path P_{uv} with endpoints $\phi(u), \phi(v)$,

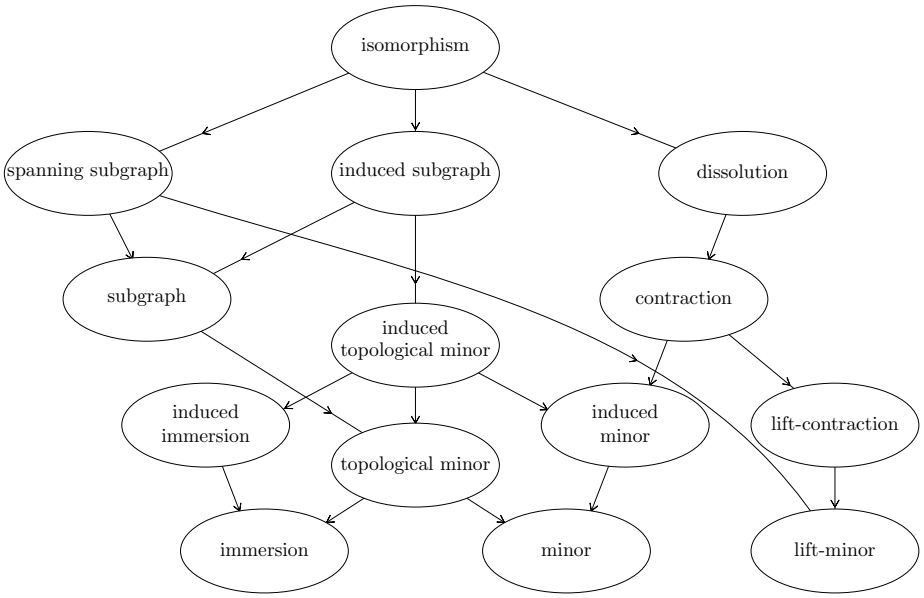


Figure 3.5: Hierarchy of containment relations. An arrow pointing from relation $\leq_1$ to relation $\leq_2$ means that for every two graphs G and H, if $H \leq_1 G$ then $H \leq_2 G$.

- for every two distinct edges $e, e' \in E(H)$, the paths P_e and $P_{e'}$ are edge-disjoint.

Since G is subcubic, we have that for any two edges $e, e' \in E(H)$, the paths P_e and $P_{e'}$ are vertex-disjoint. Therefore we can apply Proposition 3.6 and conclude that G contains H as a topological minor as well. □

Proposition 3.11. *Let G and H be two graphs such that H is subcubic. G contains H as a minor if and only if G contains H as a topological minor.*

Proof. If G contains H as a topological minor, then G clearly contains H as a minor as well, hence we only have to prove the converse statement. Assume that G contains H as a minor. By Proposition 3.3, G has an H-minor witness structure $\mathcal{W}$. We show that there is a subgraph G' of G such that G' is subcubic and H is a dissolution of G'. We construct the graph G' and an H-witness structure $\mathcal{W}'$ of G' as follows. Recall that, for every vertex $x \in V(H)$, there are at most 3 vertices adjacent to x in H. If $N_H(x) = \{y_1\}$, we keep a unique vertex v_1 in $W(x)$ that is adjacent to at least one vertex of $W(y_1)$, and delete all other vertices

of $W(x)$, and we set $W'(x) = \{v_1\}$. If $N_H(x) = \{y_1, y_2\}$ with $y_1 \neq y_2$, we find two not necessarily distinct vertices v_1 and v_2 in $W(x)$ such that $N(v_1) \cap W(y_1) \neq \emptyset$ and $N(v_2) \cap W(y_2) \neq \emptyset$, and a shortest path P from v_1 to v_2 in $W(x)$. We delete all vertices of $W(x)$ other than the vertices of P and set $W(x) = V(P)$. In the last case where $N_H(x)$ contains 3 distinct vertices y_1, y_2, y_3, we select 3 vertices v_1, v_2, v_3 such that $N(v_1) \cap W(y_1) \neq \emptyset$, $N(v_2) \cap W(y_2) \neq \emptyset$ and $N(v_3) \cap W(y_3) \neq \emptyset$. We also choose a tree T in $W(x)$ such that $\{v_1, v_2, v_3\} \subseteq V(T)$ and there is no subtree T' of T with that property. Observe that the set of leaves L of T satisfies $L \subseteq \{v_1, v_2, v_3\}$, and hence T is subcubic with at most one internal vertex of degree at least 3. Moreover, the vertices v_1, v_2 and v_3 have degree at most 2 in T, and at most one of them has degree exactly 2. We can now delete all vertices of $W(x)$ other than those of T and set $W(x) = V(T)$. Finally, for every edge $xy \in E(H)$, we arbitrarily choose two vertices $u \in W(x), v \in W(y)$ and delete all edges $\{u'v' \mid u' \in W(x), v' \in W(y), u'v' \neq uv\}$. Observe that, by construction of $\mathcal{W}'$, every set $W'(x)$ is connected and contains at most one vertex of degree 3, and no vertex of degree more than 3. Moreover, the vertices of $W(x)$ that have neighbors that do not belong to $W(x)$ all have at most 2 neighbors inside, and a unique neighbor outside of $W(x)$. Therefore, H is a dissolution of G': for every set $W(x)$, let v_x be a vertex of maximum degree in $W(x)$. Note that every vertex in $W(x) \setminus \{v_x\}$ has degree at most 2 and can therefore be dissolved. Applying this procedure for every vertex $x \in V(H)$ yields a graph G'' in which every vertex v_x satisfies $N_{G''}(v_x) = \{v_y \mid y \in N_H(x)\}$, which implies that G'' and H are isomorphic, hence completing the proof of Proposition 3.11. □

The following proposition exhibits an important relation between contraction and topological minor on planar graphs. This relation is used in [87] to devise an algorithm that decides in polynomial time if an input graph G contains a fixed graph H as a contraction when G has bounded genus. First, we need the following definition:

Definition 3.12 (Thin graph, see [2])**.** *A planar graph $G = (V, E)$ with multiple edges is thin if there exists a planar embedding such that no two multiple edges are homotopic. This means that if there are two edges e_1, e_2 between a pair of distinct vertices $v, w \in V$, then there must be two further vertices $u_1, u_2 \in V$ that sit inside the two disjoint areas of the plane that are enclosed by e_1, e_2.*

Proposition 3.13 (Lemma 2 in [87])**.** *Let G, H be thin graphs and G^*, H^**

their respective duals. Then G contains H as an embedded contraction if and only if G^* contains H^* as an embedded topological minor.

Finally, the lemma below provides a link between induced and non-induced relations. It is based on the observation that, roughly speaking, if we subdivide every edge in a graph, then deleting a vertex w obtained by subdividing edge uv is equivalent to deleting uv in the original graph. This observation was used in particular by Matoušek and Thomas [113] to derive NP-completeness results for the problems of deciding, given two input graphs G and H, if G contains H as an induced minor, induced topological minor or induced subgraph from NP-completeness of the decision problem associated with the respective non-induced counterpart of each of these relations. The complexity of deciding containment relations is discussed in detail in the next section.

Lemma 3.14 (see e.g., [113])**.** *Let G and H be two graphs, and G^* and H^* be the graphs obtained from G and H, respectively, by subdividing every edge exactly once. The following holds:*

- *G contains H as a minor if and only if G^* contains H^* as an induced minor;*
- *G contains H as a topological minor if and only if G^* contains H^* as an induced topological minor;*
- *G contains H as a subgraph if and only if G^* contains H^* as an induced subgraph.*

Proof. For the forward direction, it is sufficient to observe that if G contains H as a subgraph (respectively minor, topological minor) then G^* contains H^* as an induced subgraph (respectively induced minor, induced topological minor), since the deletion of an edge e in G can be performed in G^* by deleting the vertex resulting from the subdivision of e in G^*.

For the backward direction, let us first consider the case where G^* contains H^* as an induced minor. The case where H and H^* are either paths or cycles can immediately be observed to hold, so we assume that all of H, H^*, G and G^* have a vertex of degree at least 3. We call *blue* vertices the vertices of G^* that are also vertices of G, and *red* vertices those obtained by subdividing every edge of G. We define blue and red vertices of H^* in a similar way. We show how to construct an H-minor witness structure $\mathcal{W}'$ of G from an H^*-induced minor witness structure $\mathcal{W}$ of G^*. Our first step is to alter $\mathcal{W}$ in such a way that for every blue vertex u of H, the set $W(u)$ contains at least one blue vertex of G^*. We do this

by finding, for every blue vertex u of H^* such that $W(u)$ contains only a single red vertex, a path P in H^* whose endpoints are u and a blue vertex v such that $W(v)$ contains at least one blue vertex. Note that P consists of vertices of alternating color, starting with a red vertex and ending with a blue vertex. Moreover, the vertex v necessarily exists, since H^* contains a vertex degree at least 3, whose witness set in $\mathcal{W}$ must contain at least one blue vertex. We denote by $x_1, \ldots, x_{|V(P)|}$ the vertices of P, with $x_0 = u$ and $x_{|V(P)|} = v$, and we set $W(x_{2i-1}) := W(x_{2i-1}) \cup W(x_{2i})$, for every $1 \leq i \leq |V(P)| - 1$. Observe that this process strictly decreases the number of blue vertices of H^* whose witness sets contain only a unique red vertex. Hence, at the end of this process, the witness set of every blue vertex in H^* contains at least one blue vertex. Our next step is to merge the witness set of every red vertex of H^* with the witness set of one of its two (blue) neighbors, chosen arbitrarily. Finally, we dissolve every red vertex in G^*. We claim that this yields an H-minor witness structure $\mathcal{W}'$ of G. First, the fact that the witness sets of $\mathcal{W}'$ are connected follows from the observation that every witness set of $\mathcal{W}'$ is the union of witness sets of $\mathcal{W}$ that correspond to adjacent vertices in H^*. The fact that for every edge $uv \in E(H)$, there are vertices $x \in W'(u), y \in W'(v)$ such that $xy \in E(G)$ follows from the observation that the red vertex of G^* obtained by subdividing xy was added to either $W(u)$ or $W(v)$ before being dissolved, and hence there are two adjacent vertices G such that one belongs to $W'(u)$ and the other $W'(v)$. Thus $\mathcal{W}'$ is indeed an H-minor witness structure of G, and the proposition holds. The proofs of the cases for induced topological minor and induced subgraph are similar. □

3.4 Complexity of deciding containment relations

In this section we give an overview of the complexity of the decision problems associated with the containment relations presented in Section 3.1. For each of the relations, the name of the relation in capital letters will refer to the decision problem associated with it. For example, the decisions problem associated with the minor relation is defined as follows:

MINOR
Instance: A graph G.
Question: Does G contain H as a minor?

As we will see in more detail later, the decision problems associated with almost all the containment relations we consider are NP-complete

when both G and H are part of the input. We will actually see that this remains true even if strong restrictions are put on the structure of both G and H (see Section 3.7). Therefore, we naturally consider the cases where the pattern H is either a fixed graph not part of the input, or where the size of H is used as a parameter of the problem. As discussed in Chapter 2, we add H as a prefix to the problem name in such cases. For example, the decision problem associated with the minor relation parameterized by $|V(H)|$ is defined as follows:

H-Minor
Instance: A graph G.
Parameter: $|V(H)|$.
Question: Does G contain H as a minor?

The other problems are defined analogously for parameterized or fixed H. We are now ready to give the full list of problems we consider, in their non-parameterized form:

- Minor for the minor relation;
- Induced Minor for the induced minor relation;
- Topological Minor for the topological minor relation;
- Induced Topological Minor for the induced topological minor relation;
- Immersion for the immersion relation;
- Induced Immersion for the induced immersion relation;
- Contractibility for the contraction relation;
- Dissolution for the dissolution relation;
- Lift-contraction for the lift-contraction relation;
- Lift-minor for the lift-minor relation;
- Subgraph Isomorphism for the subgraph relation;
- Induced Subgraph Isomorphism for the induced subgraph relation;

- Spanning Subgraph Isomorphism for the spanning subgraph relation;
- Isomorphism for the isomorphism relation;

Note that the Topological Minor problem is often referred to as the Subgraph Homeomorphism problem, as e.g., in [113].

We split the overview given in this section into 2 parts: first when the target graph H is part of the input, then when it is either fixed or a parameter of the problem. Note that G is always a part of the input.

3.4.1 When H is part of the input

We consider problems associated with containment relations when H is part of the input and in their non-parameterized form. The complexity status is overall rather simple, as observed by Matoušek and Thomas [113]: all the problems are NP-complete, with the exception of Graph Isomorphism which lies in its own complexity class, as it remains a notorious open problem. Note that only the following problems were considered in [113]: Minor, Induced Minor, Contractibility, Topological Minor, Induced Topological Minor, Dissolution, Subgraph Isomorphism, Induced Subgraph Isomorphism and Graph Isomorphism. Therefore, the problems Immersion, Induced Immersion, Lift-contraction, Lift-minor and Spanning Subgraph Isomorphism also have to be considered. However, considering that Matoušek and Thomas only observed NP-completeness of the aforementioned problems, we provide explicit proofs of NP-completeness for all 14 problems of Table 3.1, with the exception of Graph Isomorphism, which is believed not to be NP-complete, and Induced Immersion, whose NP-completeness proof is given in Chapter 8. We start with the following simple proposition, based on an observation of Damashke [36].

Proposition 3.15. *The* Minor, Topological Minor, Immersion, Subgraph Isomorphism, Induced Immersion *and* Spanning Subgraph Isomorphism *problems are* NP*-complete, even if G and H are disjoint unions of paths.*

Proof. We give the proof for the Minor problem only as it is a simple task to adapt it for the other problems. We reduce from the 3-Partition problem, which is well-known to be NP-complete in the strong sense (see e.g., [63]). The problem is as follows:

3-Partition
Instance: An integer B and a family of integers S with $|S| = 3m$, for some integer m, and $B/4 < x < B/2$ for every $x \in S$.
Question: Can S be partitioned into sets $S_1, \ldots, S_m$ such that $\sum_{x \in S_i} x = B$ for every $1 \leq i \leq m$?

Given an instance I of 3-Partition, we construct an instance of Minor as follows. For each element $x \in S$, we create a path P_x on x vertices, and set H to be the disjoint union of all the paths P_x. Finally, we set G to be the graph mP_B, i.e., the disjoint union of m paths on B vertices. We claim that G contains H as a minor if and only if I is a yes-instance of 3-Partition. If I is a yes-instance of 3-Partition, then G can easily be seen to contain H as a minor by assigning one of the paths P_B of G to each set S_i of the solution of I, and deleting exactly 2 edges of it to form 3 paths of the desired lengths, one for each element of S_i. For the other direction, assume that G contains H as a minor. Observe first that $|V(G)| = |V(H)| = mB$, and hence H can be obtained from G by performing only edge deletions. Hence there is a partition $\mathcal{P}$ of the paths of H such that every element P of $\mathcal{P}$ is associated with a path of length B of G, and the sum of the length of the elements of P is equal to B. Since every path of H is associated with an element x of S in I, we conclude that there is a partition of S into B sets whose elements sum up to B, i.e., I is a yes-instance of 3-Partition. □

Note that in the proof of Proposition 3.15, we need to take the paths in G to have $B + 2$ vertices instead of B for the case where we consider the Induced Immersion problem. This is due to the fact that induced immersion does not allow edge deletion, and hence H is obtained from G by deleting vertices. We would also like to point out that the case where H is a cycle can be shown to be NP-complete by a straightforward reduction from the Hamiltonian Cycle problem.

An interesting corollary of Proposition 3.15, obtained by combining it with Lemma 3.14, is the following.

Proposition 3.16. *The* Induced Minor, Induced Topological Minor *and* Induced Subgraph Isomorphism *problems are* NP*-complete, even if G and H are disjoint unions of paths.*

We now move on to the next problem: Dissolution.

Theorem 3.17 (See e.g., [113])**.** *The* Dissolution *problem on simple graphs is* NP*-complete.*

Proof. We reduce from the Hamiltonian Cycle problem, which is well-known to be NP-complete. Given an instance G' of Hamiltonian Cycle, we construct two graphs G and H such that H is a dissolution of G if and only if G' contains a Hamiltonian cycle. The graph G consists of a clique C with vertex set $V(G')$, and for every edge $uv \in E(G')$ we add a vertex x_{uv} that we make adjacent with u and v only. The graph H consists of a cycle with vertex set $\{z_1, \ldots, z_{2|V(G')|}\}$ where the vertices z_i such that i is even are made into a clique. We now show that H is a dissolution of G if and only if G' contains a Hamiltonian cycle.

If G' contains a Hamiltonian cycle with edges E', then we dissolve all the vertices x_{uv} such that $uv \notin E$. This clearly yields a graph isomorphic to H, proving that H is a dissolution of G.

For the converse direction, assume that H is a dissolution of G. Observe that only vertices of degree 2 can be dissolved, and that the only such vertices in G are $x_{uv}, uv \in E(G')$. Moreover, the set of vertices x_{uv} of G that are not dissolved in order to obtain H can be seen to correspond to a set of edges in G' that forms a cycle of length $|V(G')|$, i.e., a Hamiltonian cycle in G'. This concludes the proof of Theorem 3.17. □

Finally, out of the 14 problems associated with the containment relations presented in Table 3.1, we have left Lift-contraction, Lift-minor and Graph Isomorphism.

The computational complexity of the Graph Isomorphism problem is a famous long-standing open problem, and it is not known whether it admits a polynomial time algorithm or not. However, it was shown that Graph Isomorphism is not NP-complete unless the polynomial hierarchy collapses to a finite level [137]. This lead to the notion of Graph Isomorphism-complete (GI-complete for short) problems, that is the class of problems that have a polynomial time reduction to and from Graph Isomorphism.

As for the Lift-contraction and Lift-minor problems, they were introduced very recently by Golovach, Kamiński, Paulusma and Thilikos in [71] and studied only from a combinatorial point of view. However, it is not very difficult to show that both problems are NP-complete.

Proposition 3.18. *The* Lift-contraction *and* Lift-minor *problems are* NP*-complete, even if G is a planar graph of maximum degree at most 4 and H is a cycle.*

Proof. We reduce from the Hamiltonian Cycle problem on 3-regular planar graphs, which is known to be NP-complete [64]. Given a 3-regular

planar graph $G = (V, E)$ on n vertices, we construct a graph G' by adding, for every vertex $v \in V$, a new vertex v' that is made adjacent to v only. We refer to the newly added vertices as *red* vertices, and to the other vertices of G' as *black* vertices. Any edge incident with a red vertex is called a red edge, and any other edge is black. We claim that G has a Hamiltonian cycle if and only if G' contains C_n as a lift-minor if and only if G' contains C_n as a lift-contraction.

First suppose G has a Hamiltonian cycle C, and let C' be the corresponding cycle in G'. In G', we lift every black edge uv that does not belong to C' with a red edge incident to either u or v. By construction, such a red edge will always be available. Afterwards, we contract every remaining red edge. This shows that we can obtain C' from G' by a sequence of edge contractions and lifts, which implies that G' contains C_n as a lift-contraction, and hence as a lift-minor.

For the reverse direction, suppose G' contains C_n as a lift-minor. Note that every red vertex has degree 1 in G', and will also have degree 1 in any lift-minor G'' of G, as lifts, edge deletions and edge contractions do not increase the degree of vertices of degree 1. Hence any cycle in G'' contains only black vertices. Therefore, if there is a cycle C'' of length n in G'', C'' contains all the black vertices. Since there are exactly n black vertices in G, we conclude that no black edge of G is contracted in order to obtain G''. Let us consider the sequence of lifts $\mathcal{L}$ applied to G in order to obtain G''. Since lifts do not increase the degree of any vertex and red vertices have degree 1 in G', no red vertex is incident to both edges of any lift in $\mathcal{L}$. This implies that we cannot create a black edge by lifting two red edges. We can also not create a black edge by lifting a red edge with a black edge, as such a lift results in a an edge incident with a red vertex, and such an edge is red by definition. This means that we can only create a black edge by lifting two black edges. Consequently, we cannot increase the number of black edges incident with any vertex. Now suppose, for contradiction, that there is an edge uv in C'' that was not present in G'. Since all the edges in C'' are black, the edge uv must have been created by lifting two black edges $\{uw, wv\}$. Since w was incident with 3 black edges in G', such a lift decreases the number of black edges incident with w down to 1. This contradicts the fact that w is contained in the cycle C'', all whose edges are black. We conclude that C'' was already present as a subgraph in G', which implies that G has a Hamiltonian cycle. □

3.4.2 When H is fixed

Now, we turn our attention to the cases where H is either fixed or a parameter of the instance. In these two cases, the diversity of results and techniques involved to show either tractability or intractability is much greater than when H is part of the input and not a parameter. This can most adequately be observed in Table 3.2 below.

Containment Problem	Complexity with parameter $\|V(H)\|$
H-Minor	$O(\|V(G)\|^3)$ [129]
H-Induced Minor	NP-complete for fixed H [55]
H-Topological Minor	$O(\|V(G)\|^3)$ [79]
H-Induced Topological Minor	NP-complete for fixed H [105]
H-Immersion	$O(\|V(G)\|^3)$ [79]
H-Induced immersion	In XP (Chapter 8, [12])
H-Contractibility	NP-complete for fixed H [23]
H-Lift-contraction	Open [71]
H-Lift-minor	Open [71]
H-Dissolution	$O(\|V(G)\|^3)$ [72]
H-Subgraph Isomorphism	In XP
H-Induced Subgraph Isomorphism	In XP
H-Spanning Subgraph Isomorphism	$O(1)$
H-Graph Isomorphism	$O(1)$

Table 3.2: The parameterized complexity of the 14 problems from Table 3.1 on general graphs.

Tractable cases

In [129], Robertson and Seymour gave an algorithm that solves the H-Minor problem in time $O(|V(G)|^3)$. Recently, Kawarabayashi, Kobayashi and Reed [94] gave a new algorithm with running-time $O(|V(G)|^2)$. Since both algorithms are provided together with polynomial-time algorithms for the k-Disjoint Paths (and k-Edge-Disjoint Paths) problem parameterized by the number of pairs of terminals k, it follows that the H-Topological Minor and H-Immersion problems can also be solved in polynomial time for every fixed graph H (see Section 3.5). However, the algorithms for both problems run in time $|V(G)|^{f(|V(H)|)}$ for some function f. The question whether H-Topological Minor and H-Immersion belong to FPT when parameterized by $|V(H)|+|E(H)|$ remained open until recently, when Grohe, Kawarabayashi, Marx and Wollan [79] provided $O(|V(G)|^3)$ time algorithms for both problems.

The H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism problems are easily seen to solvable in polynomial time for every graph H by brute-force with the following observation:

- H is an induced subgraph of H if and only if H is isomorphic an induced subgraph of G on $|V(H)|$ vertices;
- H is a subgraph of H if and only if H is a spanning subgraph of an induced subgraph of G on $|V(H)|$ vertices.

Moreover, the H-Spanning Subgraph Isomorphism and H-Graph Isomorphism problems are easily seen to be solvable in constant time for every fixed graph H, since in these problems the graph G has the same number of vertices as H, and therefore solving the problem by brute-force on G is sufficient.

Finally, we consider the case of H-Dissolution. The problem is easily seen to be in FPT, and a proof was provided by Golovach, Kamiński, Paulusma and Thilikos [72]. We also give a proof here for completeness, although with slightly worse constants.

Theorem 3.19. *The H-Dissolution problem admits a vertex-kernel of size $O(|V(H)|^3)$.*

Proof. Given graphs G and H, we apply the following reduction rules to G:

(i) if G has at least $|V(H)|+1$ vertices of degree 3 or more, then output a trivial no-instance, e.g., a complete graph on $|V(H)|+1$ vertices;

(ii) if there exist vertices $u_1, \ldots, u_{|V(H)+1}$ such that $d_G(u_i) = 2$ for every $1 \leq i \leq |V(H)|$ and $u_1u_{i+1} \in E(G)$ for every $1 \leq i \leq |V(H)|-1$, then dissolve u_1.

It is clear that applying these two rules yields a graph that has at most $|V(H)|$ vertices of degree 3 or more, and every 2 such vertices are possibly joined by a path of at most $|V(H)|$ vertices of degree 2. Therefore the graph we obtain after applying the reduction rules has at most $O(|V(H)|^3)$ vertices.

We now show that both reduction rules are sound. The fact that rule (i) is sound simply follows from the fact that vertices of degree at least 3 cannot be dissolved, and therefore every dissolution of G contains at least as many vertices of degree 3 as G, and H obviously contains at most $|V(H)|$ vertices of degree at least 3. Soundness of rule (ii) follows from

the fact that at least one vertex of $u_1, \ldots, u_{|V(H)|+1}$ must be dissolved, as otherwise there would be at least $|V(H)| + 1$ vertices left. Moreover, assume that $u_i, i \neq 1$, is to be dissolved in order to yield H. Observe that $G \wedge \{u_i\}$ is isomorphic to $G \wedge \{u_1\}$. Hence we can dissolve u_1 instead, which shows that rule (ii) is sound. Together with the fact that both rules can be applied in polynomial time, this completes the proof of Theorem 3.19. □

Finally, the following was recently proved by Belmonte, van 't Hof and Kamiński in [12] (see Chapter 8):

Theorem 3.20 ([12]). *The* H-Induced Immersion *can be solved in polynomial time for every graph* H.

This can appear somewhat surprising if we compare it with the situation for H-Induced Minor and H-Induced Topological Minor, which are both NP-complete for some specific fixed graphs (see Theorems 3.21 and 3.22 below).

Intractable cases

As suggested by Lemma 3.14, induced containment relations tend to be somewhat harder to decide than their non-induced counterparts. This is especially true in the case of the H-Induced Minor and H-Induced Topological Minor problems. While H-Minor and H-Topological Minor admit quadratic FPT algorithms, as mentioned previously, H-Induced Minor and H-Induced Topological Minor are not even solvable in polynomial time for every fixed graph H. Fellows, Kratochvíl, Middendorf and Pfeiffer [55] showed in particular that there exists a graph H^* on 68 vertices for which H^*-Induced Minor is NP-complete; the graph H^* is depicted in Figure 3.6. As for H-Induced Topological Minor, it was proved by Lévêque, Lin, Maffray and Trotignon [105] that K_5-Induced Topological Minor is NP-complete.

Theorem 3.21 ([55]). *The* H-Induced Minor *problem is* NP*-complete when* H *is the graph depicted in Figure 3.6.*

Theorem 3.22 ([105]). *The* K_5-Induced Topological Minor *problem is* NP*-complete.*

Moreover, it is interesting to compare the H-Contractibility problem with H-Dissolution, since they allow seemingly similar operations. While the latter is among the most efficiently solvable relations we considered, this spectacularly fails to be true for H-Contractibility. Indeed,

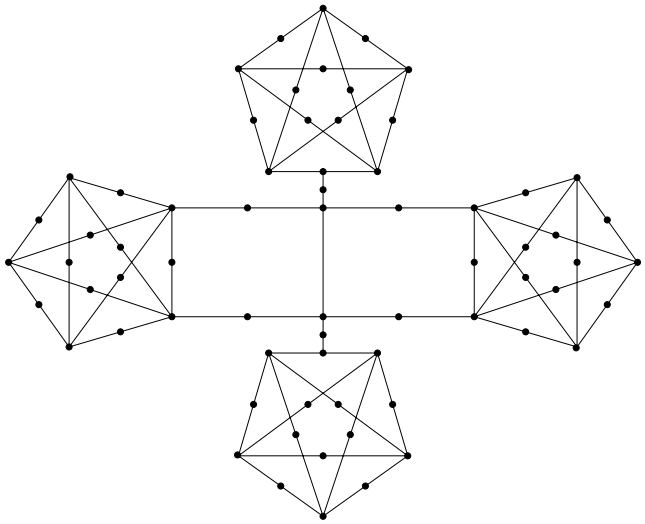


Figure 3.6: Graph H for which H-Induced Minor is NP-complete [55].

Brouwer and Veldman proved in [23] that H-Contractibility is already NP-complete when H is isomorphic to either P_4 or C_4.

Theorem 3.23 ([23])**.** *The two problems* P_4-Contractibility *and* C_4-Contractibility *are* NP*-complete.*

Brouwer and Veldman used a reduction from the Hypergraph 2-Coloring problem and the construction depicted in Figure 3.7. This construction is frequently adapted in order to prove NP-completeness for various problems related to H-Contractibility, see e.g., [9].

We mentioned previously that H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism can be easily solved in polynomial time for every fixed graph H by a simple brute-force approach. It is then natural to ask if it is possible to obtain a more efficient algorithm for these two problems. This appears not to be the case, because of the following.

Theorem 3.24 (See [48])**.** *The* k-Clique *problem is* W[1]*-complete.*

Observe indeed that the k-Clique problem is equivalent to both K_k-Subgraph Isomorphism and K_k-Induced Subgraph Isomorphism, which immediately implies W[1]-hardness for both H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism.

Corollary 3.25. *The two problems* H-Subgraph Isomorphism *and* H-Induced Subgraph Isomorphism *are* W[1]*-hard when parameterized by* $|V(H)|$.

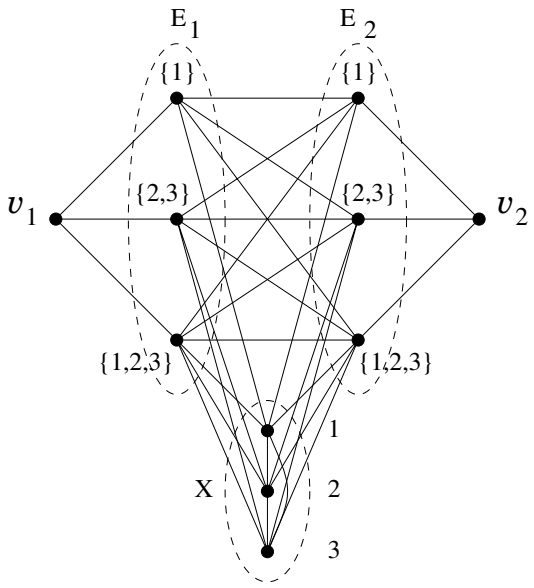


Figure 3.7: Example of the construction used by Brouwer and Veldman [23] to show that P_4-Contractibility is NP-complete. They give a reduction from the NP-complete problem Hypergraph 2-Coloring, which takes as input a ground set X and a family of sets E and asks if there is a partition (X_1, X_2) of X such that every set in E contains an element of X_1 and an element of X_2.

Finally, we conclude with the H-Lift-contraction and H-Lift-minor problems. As mentioned previously, these two problems have not yet been studied from an algorithmic point of view, and therefore their complexity status is still open.

3.5 Containment relations and disjoint paths

In this section, we describe algorithms to prove that the problems H-Minor, H-Topological Minor and H-Immersion belong to XP, i.e., we give algorithms whose running-time is polynomial for every fixed graph H. These three problems in fact belong to FPT (see [129] for H-Minor and [79] for H-Topological Minor and H-Immersion), but those algorithms are much more involved. There are two motivations to give these XP algorithms here. First, they allow us to illustrate the important relation between these problems and the k-Disjoint Paths problem. Second, the ideas used in these algorithms, however simple, allow to design effi-

cient algorithms in other cases, such as in [7], where similar ideas are used to solve H-Induced Minor and H-Contractibility on chordal graphs.

k-Disjoint Paths
Instance: A graph G and k pairs $(s_1, t_1), \ldots, (s_k, t_k)$ of vertices of G, called the *terminals*,
Parameter: The number of terminal pairs k,
Question: Does G contain k pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i has endpoints s_i and t_i for every $1 \leq i \leq k$?

If we ask for pairwise edge-disjoint paths instead of vertex-disjoint, we call the problem k-Edge-Disjoint Paths. The Disjoint Paths problem, when the number of pairs of terminals k is part of the input, was first shown to be NP-complete by Karp [90]. When the number of pairs of terminals is fixed, it was shown that the problem can be solved in polynomial time when $k = 2$ on simple graphs [139, 141, 145] but is NP-complete on directed graphs even when restricted to the case where $k = 2$ [61]. In [129], Robertson and Seymour provided an algorithm for the k-Disjoint Paths problem that runs in time $O(|V(G)|^3)$ for any input graph G. The running time of the algorithm was later improved to $O(|V(G)|^2)$ by Kawarabayashi, Kobayashi and Reed in [94], where they also provide an algorithm for the k-Edge-Disjoint Paths problem that also runs in time $O(|V(G)|^2)$.

Theorem 3.26 ([94])**.** *The k-Disjoint Paths and k-Edge-Disjoint Paths problems can be solved in $O(|V(G)|^2)$ time.*

The following simple observation regarding the polynomial time solvability of both the H-Topological Minor and H-Immersion problems follows as a corollary of Theorem 3.26.

Proposition 3.27. *The H-Topological Minor and H-Immersion problems can be solved in polynomial time for every fixed multigraph H.*

Proof. First, consider the following algorithm for H-Topological Minor with input graph G and target multigraph H: for every subset S of $V(G)$ such that $|S| = |V(H)|$ and every bijection $\phi : V(H) \to S$, we construct an instance $I_{S,\phi}$ of the k-Disjoint Paths problem with input graph G and set of terminal pairs composed of $\text{mult}(uv)$ copies of the the pair $(\phi(u), \phi(v))$ for each pair of vertices $u, v \in V(H)$. If $I_{S,\phi}$ is a yes-instance, then G contains H as a topological minor and we output "yes", otherwise

we try another choice for the bijection ϕ. If every choice of ϕ yields a no-answer, we try a different set S of vertices. If no such choice yields a yes-answer, we conclude that G does not contain H as a topological minor and output "no". To show that the algorithm terminates within the desired running-time, it suffices to observe that there are at most $|V(G)|^{|V(H)|}$ choices for the set S and $|V(H)|!$ choices for the bijection ϕ, for each choice of S. Finally, since the instance $I_{S,\phi}$ of the k-Disjoint Paths problem can be solved in time $O(f(|E(H)|) \cdot |V(G)|^2)$ time due to Theorem 3.26, we obtain a total running-time of $O(f(|E(H)|) \cdot |V(G)|^{|V(H)|+2})$.

The algorithm for H-Immersion is identical, except for the fact that the instance $I_{S,\phi}$ that we create is an instance of the k-Edge Disjoint Paths problem. This concludes the proof of Observation 3.27. □

The idea for the two algorithms in Observation 3.27 is a simple application of the definition of the topological minor and immersion relations in terms of existence of (edge-)disjoint paths that was given in Section 3.2 (see Propositions 3.6 and 3.8). A similar idea is used in [12] to show that the H-Induced Immersion problem can be solved in polynomial time for every fixed target multigraph H, although additional issues appear due to the possibility of having "unwanted" edges (see Chapter 8 for more details).

We now show how to solve the H-Minor problem in polynomial time for every fixed graph H. As with topological minor and immersion, we want to use the solution of the k-Disjoint Paths problem as a subroutine to help us solve the problem, however the link between minors and disjoint paths is somewhat less obvious than for topological minors and immersions. In order to observe this connection, consider the following problem:

t-Disjoint Connected Subgraphs
Instance: A graph G and k pairwise disjoint nonempty vertex subsets $S_1, \ldots, S_k$ of G,
Parameter: The total number of terminals $t = \sum_{i=1}^{k} |S_i|$,
Question: Does G contain k pairwise vertex-disjoint subgraphs $G_1, \ldots, G_k$ such that $S_i \subseteq V(G_i)$, for every $1 \leq i \leq k$?

The connection between H-Minor and t-Disjoint Connected Subgraphs, where $t = \sum_{i=1}^{k} |S_i|$, follows from Definition 3.2. We now need to show how to solve t-Disjoint Connected Subgraphs via k-Disjoint Paths.

Lemma 3.28. *The t-Disjoint Connected Subgraphs problem can be solved in $|V(G)|^{O(t)}$ time.*

Proof. Let $S_1, \ldots, S_k$ be the sets of terminals of the instance of t-Disjoint Connected Subgraphs. Observe first that we only need to consider the case where the subgraphs $G_1, \ldots, G_k$ are trees. This can be seen from the fact that any solution $G_1, \ldots, G_k$ consisting of arbitrary subgraphs of G can be turned into another solution $T_1, \ldots, T_k$ such that every T_i is a tree. This can be done by taking T_i to be any spanning subgraph of G_i, for every $1 \leq i \leq k$.

Now, for each of these sets S_i, we consider a set S'_i of vertices that will form the branch vertices of the tree T_i. Note that, without loss of generality, we may consider the case where every leaf of T_i is a terminal of S_i, since for every non-terminal leaf $u \in T_i$, the subtree $T_i - u$ forms a connected subgraph of G, contains every vertex of S_i and is vertex-disjoint from every other tree $T_j, 1 \leq j \neq i \leq k$. Hence, we obtain that $|S'_i| \leq |S_i| - 2$, since the number of branch vertices in a tree is no more than its number of leaves minus 2.

Given the sets S_i and S'_i, we now consider every possible tree $\mathcal{T}_i$ on vertex set $S_i \cup S'_i$, for every $1 \leq i \leq k$. For each choice of the trees $\mathcal{T}_1, \ldots, \mathcal{T}_k$, we denote by $\mathcal{F}$ the disjoint union of the trees $\mathcal{T}_1, \ldots, \mathcal{T}_k$. Given the forest $\mathcal{F}$, we construct a instance $I_{\mathcal{F}}$ of k-Disjoint Paths by making every pair of adjacent vertices of $\mathcal{F}$ into a pair of terminals (s_j, t_j) of $I_{\mathcal{F}}$.

Claim 1. The sets $S_1, \ldots, S_k$ in G form a yes-instance of t-Disjoint Connected Subgraphs if and only if there exists a forest $\mathcal{F}$ such that $I_{\mathcal{F}}$ is a yes-instance of k-Disjoint Paths.

Proof of Claim 1. For the forward direction, assume that there exist trees $T_1, \ldots, T_k$ that are pairwise vertex-disjoint and such that $S_i \subseteq V(T_i)$, for every $1 \leq i \leq k$. We choose $\mathcal{F}$ as follows:

- for every tree $T_i, 1 \leq i \leq k$, the set of vertices S'_i consists of every branch vertex of $T_i - S_i$;
- for every pair of vertices $u, v \in S_i \cup S'_i$, we add an edge between u and v in $\mathcal{T}_i$ if and only if there is a path (possibly of length 1) in T_i between u and v whose inner vertices all have degree 2 in T_i.

It can be observed that every connected component of $\mathcal{F}$ is formed by the vertices of $S_i \cup S'_i$. Moreover, $\mathcal{F}$ is a forest, as any cycle in $\mathcal{F}$ would correspond to a cycle in some tree $T_i, 1 \leq i \leq k$. Therefore we conclude that if $S_1, \ldots, S_k$ form a yes-instance of t-Disjoint Connected Subgraphs

in G, then we can construct a forest $\mathcal{F}$ such that $I_{\mathcal{F}}$ is yes-instance of k-Disjoint Paths.

For the backward direction, assume that there exists a forest $\mathcal{F}$ such that:

- $\mathcal{F}$ has connected components $\mathcal{T}_1, \ldots, \mathcal{T}_k$ and $S_i \subseteq \mathcal{T}_i$ for every $1 \leq i \leq k$;
- for every pair of vertices $u, v \in V(\mathcal{F})$ such that $uv \in E(\mathcal{F})$, there is a path P_{uv} in G and for every two pairs of vertices u, v and u', v', the paths $P_{u,v}$ and $P_{u',v'}$ are internally vertex-disjoint.

We claim that we can construct the trees $T_1, \ldots, T_k$ that connect terminals $S_1, \ldots, S_k$ respectively. Consider a connected component $\mathcal{T}_i$ of $\mathcal{F}$ and recall that $S_i \subseteq V(\mathcal{T}_i)$, for every $1 \leq i \leq k$. We take as T_i the graph formed by the union of the paths $P_{u,v}$, for every pair of adjacent vertices $u, v \in V(\mathcal{T}_i)$. Observe that since $\mathcal{T}_i$ is a tree and every edge of $\mathcal{T}_i$ corresponds to a path in T_i, then T_i is a tree as well. More precisely, T_i is a subdivision of $\mathcal{T}_i$. Moreover, since the paths $P_{u,v}$ are pairwise vertex-disjoint, it follows that the trees T_i are also pairwise vertex-disjoint. This concludes the proof of Claim 1. ◇

We now only have to show that the instance $I_{\mathcal{F}}$ can be found and solved in time $|V(G)|^{O(t)}$. In order to find the forest $\mathcal{F}$, we have to try every possible family of sets $S_1, \ldots, S_k$, every possible family $S'_1, \ldots, S'_k$ and every possible forest on vertex set $\bigcup_{i=1}^{k}(S_i \cup S'_i)$. Since $\sum_{i=1}^{k} |S_i| \leq t$ and $|S'_i| \leq |S_i| - 2$ for every $1 \leq i \leq k$, it follows that $\sum_{i=1}^{k} |S_i \cup S'_i| \leq 2t$, and hence there are at most $|V(G)|^{2t}$ choices for $S_i \cup S'_i$. Moreover, for every choice of $S_i \cup S'_i$, there are at most $2t^{4t}$ possible forests, since there are at most l^{2l} forests on l vertices. Hence we have at most $2t^{4t} \cdot |V(G)|^{2t}$ possible choices for the instance $I_{\mathcal{F}}$. Finally, since the k-Disjoint Paths problem can be solved in $O(|V(G)|^2)$ time for every fixed value of k, it follows that we can test whether $I_{\mathcal{F}}$ is a yes-instance of k-Disjoint Paths in time $f(t) \cdot |V(G)|^2$, for some function f. Hence we can solve the t-Disjoint Connected Subgraph problem in time $O(|V(G)|^{2t+2})$. This concludes the proof of Lemma 3.28. □

We are now ready to give an algorithm that solves the H-Minor problem in polynomial time for every fixed pattern graph H.

Proposition 3.29. *For every fixed graph H, the H-Minor problem can be solved in time $|V(G)|^{O(|V(H)|^2)}$.*

Proof. We claim that the following algorithm solves the H-Minor problem within the desired running-time. Let $S_1, \ldots, S_{|V(H)|}$ be sets of vertices with $|S_i| \leq |V(H)|$ for every $1 \leq i \leq k$, and ϕ a bijection from $V(H)$ to $\{S_1, \ldots, S_{|V(H)|}\}$ such that for every pair of vertices $u, v \in V(H)$, if $uv \in E(H)$, then there exist vertices $x \in \phi(u), y \in \phi(v)$ such that $xy \in E(G)$. It can be observed that if $S_1, \ldots, S_k$ is a yes-instance of t-Disjoint Connected Subgraphs in G, then G contains H as a minor. The reverse direction is a direct consequence of Definition 3.2.

Finally, let us analyze the running-time of the algorithm. Observe that there are at most $|V(G)|^{|V(H)|^2}$ possible choices for the sets $S_1, \ldots, S_k$ and at most $|V(H)|!$ possible bijections ϕ for every choice of $S_1, \ldots, S_k$. Hence the algorithm solves at most $|V(H)|! \cdot |V(G)|^{|V(H)|^2}$ instances of t-Disjoint Connected Subgraphs, each of which can be solved in time $O(|V(G)|^{2 \cdot |V(H)|^2 + 2})$. Therefore, we conclude that the algorithm runs in time $O(|V(G)|^{3 \cdot |V(H)|^2 + 2})$. □

3.6 Monadic Second-Order Logic

In this section, we give MSO-formulas for several of the containment relations presented in Table 3.1. Note that although it has been known that all the relations considered in this section are expressible in monadic second-order logic, the actual formulas are seldom provided explicitly. Courcelle proved in [33] that every graph property that can be expressed in monadic second-order logic can be decided in linear time on graphs of bounded treewidth.

Theorem 3.30 ([33]). *There is an algorithm that decides for every graph G and monadic second-order sentence ϕ, whether G satisfies ϕ in time $f(\phi, tw(G)) \cdot (|V(G)| + |E(G)|)$, where f is a function and $tw(G)$ is the treewidth of G.*

Later, Courcelle, Makowsky and Rotics showed in [35] that a similar result holds when considering graphs of bounded clique-width, at the cost of forbidding quantification over sets of edges.

The interest of providing these formulas is twofold. First, the problems associated with the containment relations that can be expressed in monadic second-order logic can be decided in linear FPT time on graphs of bounded clique-width as mentioned above. These include in particular distance-hereditary graphs and series-parallel graphs, both of which contain the class of trees. The other point is that solvability on graphs of bounded treewidth is an important part of the so-called irrelevant vertex

approach which is by now an established technique in parameterized algorithms. Very roughly speaking, the irrelevant vertex approach consists in either finding a large grid as a minor and concluding that there is a vertex in it whose removal does not affect the existence of a solution, or no such large grid minor exist and therefore the graph has bounded treewidth. Obviously, concluding that the input graph has bounded treewidth is not sufficient, and the last step consist in showing that the problem that is being considered can indeed be solved on graphs of bounded treewidth.

Before we start giving MSO formulas for containment relations, we first need a formula to express that a set S of vertices induces a connected subgraph of a graph G (see e.g [34]).

$$\begin{aligned} Connected(S) = \neg(\exists X \subseteq S\ (\exists x : x \in X \wedge \exists y : y \notin X \wedge \\ \forall u, v(adj(u,v) \Rightarrow u \in X \Leftrightarrow v \in X)) \end{aligned}$$

3.6.1 Minor, induced minor and contraction

We group together the MSO expressions for the minor, induced minor and contraction relations. This is due to the fact that their MSO-expressions are based on their alternative definitions using their respective witness structures, as shown in Proposition 3.2 in Section 3.2.

Minor:

$$\begin{aligned} \phi_M(G,H) = &\exists S_1, \ldots, S_k \subseteq V(G) : \\ &(\forall 1 \leq i < j \leq k \\ &(v_i v_j \in E(H) \Rightarrow (\exists x, y : x \in S_i \wedge y \in S_j \wedge adj(x,y))))) \\ &\wedge(\forall 1 \leq i \leq k\ Connected(S_i)) \\ &\wedge(\forall 1 \leq i < j \leq k(S_i \cap S_j = \emptyset)) \end{aligned}$$

Induced minor:

$$\begin{aligned} \phi_{I.M}(G,H) = &\exists S_1, \ldots, S_k \subseteq V(G) : \\ &(\forall 1 \leq i < j \leq k \\ &(v_i v_j \in E(H) \Leftrightarrow (\exists x, y : x \in S_i \wedge y \in S_j \wedge adj(x,y))))) \\ &\wedge(\forall 1 \leq i \leq k\ Connected(S_i)) \\ &\wedge(\forall 1 \leq i < j \leq k(S_i \cap S_j = \emptyset)) \end{aligned}$$

Contraction:

$$\begin{aligned}\phi_{Con}(G,H) =& \exists S_1,\ldots,S_k \subseteq V(G): \\ & \forall x \in V(G)(\exists 1 \leq i \leq k : x \in S_i) \\ & (\forall 1 \leq i < j \leq k \\ & (v_iv_j \in E(H) \Leftrightarrow (\exists x,y : x \in S_i \wedge y \in S_j \wedge adj(x,y))))) \\ & \wedge(\forall 1 \leq i \leq k\ Connected(S_i)) \\ & \wedge(\forall 1 \leq i < j \leq k(S_i \cap S_j = \emptyset))\end{aligned}$$

3.6.2 Topological minor, induced topological minor and immersion

For similar reasons as above, we give the MSO-expressions for the topological minor, induced topological minor and immersion relations together, since these expressions rely on the alternative definitions of these relations provided in Section 3.2 (see Propositions 3.6, 3.7 and 3.8).

Topological minor:

$$\begin{aligned}\phi_{T.M}(G,H) =& \exists u_1,\ldots,u_k \in V(G) : \forall v_iv_j \in E(H) \exists P_{ij} \subseteq V(G) : \\ & Connected(P_{ij}) \wedge u_i \in P_{ij} \wedge v_i \in P_{ij} \\ & \wedge \forall 1 \leq i,i',j,j' \leq k(P_{ij} \cap P_{i'j'} = \{u_i,u_j\} \cap \{u_{i'},u_{j'}\})\end{aligned}$$

Induced topological minor:

$$\begin{aligned}\phi_{I.T.M}(G,H) =& \exists u_1,\ldots,u_k \in V(G) : \forall v_iv_j \in E(H) \exists P_{ij} \subseteq V(G) : \\ & Connected(P_{ij}) \wedge u_i \in P_{ij} \wedge v_i \in P_{ij} \wedge \\ & \forall 1 \leq i,i',j,j' \leq k(P_{ij} \cap P_{i'j'} = \{u_i,u_j\} \cap \{u_{i'},u_{j'}\}) \wedge \\ & \forall 1 \leq i,i',j,j' \leq k(\forall x \in P_{ij} \setminus \{u_i,u_j\}, y \in P_{i'j'}\} \neg adj(x,y))\end{aligned}$$

Immersion: For the particular case of immersion, we need to express connectivity for sets of edges instead of sets of vertices:

$$\begin{aligned}Connected\text{-}Edges(S) = \neg(\exists X \subseteq S\ (\exists e : e \in X \wedge \exists f.f \notin X \wedge \\ \forall e,f(inc(e,f) \Rightarrow e \in X \Leftrightarrow f \in X))\end{aligned}$$

Recall that immersion exists in two forms, namely strong and weak immersion. In the case of strong immersion, we need to specify that the paths do not contain any branch vertex other than their endpoints.

- Weak immersion:

$$\begin{aligned}\phi_{W.Im}(G,H) =& \exists u_1,\ldots,u_k \in V(G) : \forall v_iv_j \in E(H) \exists P_{ij} \subseteq E(G) : \\ & \textit{Connected-Edges}(P_{ij}) \wedge u_i \in P_{ij} \wedge v_i \in P_{ij} \\ & \wedge \forall 1 \leq i,i',j,j' \leq k (P_{ij} \cap P_{i'j'} = \emptyset)\end{aligned}$$

- Strong immersion:

$$\begin{aligned}\phi_{S.Im}(G,H) =& \exists u_1,\ldots,u_k \in V(G) : \forall v_iv_j \in E(H) \exists P_{ij} \subseteq E(G) : \\ & \textit{Connected-Edges}(P_{ij}) \wedge u_i \in P_{ij} \wedge v_i \in P_{ij} \\ & \wedge \forall 1 \leq i,i',j,j' \leq k (P_{ij} \cap P_{i'j'} = \emptyset) \\ & \forall 1 \leq i,j,l \leq k (u_l \in P_{ij} \Rightarrow (l = i \vee l = j))\end{aligned}$$

3.6.3 Subgraph and induced subgraph

The two last relations for which we give an MSO-expression are the subgraph and induced subgraph relations. An interesting point to observe is that the MSO-expressions for both relations do not use quantification over sets, and therefore these expressions are actually first-order logic expressions. The interest of this observation is particularly clear once combined with the result of Nešetřil and Ossona de Mendez [116] which states that any fixed graph property of the type $\exists X : (|X| \leq p) \wedge (G[X] \models \phi)$ may be decided in linear time for input graphs in a fixed class with bounded expansion, where ϕ is a first-order logic sentence. This implies, as proved by Nešetřil and Ossona de Mendez [116], that both H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism can be solved in linear time on any class of graphs having bounded expansion, which includes in particular planar graphs and graphs of bounded treewidth.

Subgraph:

$$\begin{aligned}\phi_{Sub}(G,H) =& \exists u_1,\ldots,u_k \in V(G) : \\ & \forall 1 \leq i,j \leq k (v_iv_j \in E(H) \Rightarrow adj(u_i,u_j))\end{aligned}$$

Induced subgraph:

$$\begin{aligned}\phi_{Ind.Sub}(G,H) =& \exists u_1,\ldots,u_k \in V(G) : \\ & \forall 1 \leq i,j \leq k (v_iv_j \in E(H) \Leftrightarrow adj(u_i,u_j))\end{aligned}$$

3.6.4 Open cases

In this section, we have provided MSO-expressions for 8 containment relations. We did not provide MSO-expressions for the dissolution, spanning

subgraph and isomorphism relations for the reason that these three relations already admit simple and efficient FPT algorithms on general graphs that do not require to take the treewidth of G into account. This greatly limits the interest of providing MSO-expressions for these relations. Finally, in the case of induced immersion, lift-contraction and lift-minor, it is an open question whether these relations can be expressed in monadic second-order logic.

3.7 Complexity on restricted input

In Section 3.4, we saw how efficiently the problems associated with the containment relations presented in Table 3.1 can be solved on general graphs when the pattern graph H is part of the input, a parameter, or fixed. In this section, we give a detailed overview of the complexity of these problems when restrictions are made on the structure of either G or H. For definitions of the various graph classes mentioned in this section, please refer to Section 2.2.

3.7.1 Restrictions on H

We start by considering the cases where restrictions are imposed on the structure of H only. Two kinds of restrictions can be made: either we look at one specific graph H, or a whole class of graphs such as paths, trees etc., to which H can belong.

In this section as well as the next one, we will mainly focus on the induced minor, induced topological minor, contraction and induced subgraph relations. The motivation for this choice is mainly that specific results on graph classes for these relations are much more common, because for some of the other relations, the problems associated with them can already be solved efficiently on general graphs (e.g., minor, topological minor, immersion, etc.), or because the relation has not received as much attention (e.g., induced immersion, lift-contraction, etc.).

Induced Minor: As we mentioned earlier, Fellows, Kratochvíl, Middendorf and Pfeiffer showed in [55] that there exists a graph H^* on 68 vertices for which the H^*-Induced Minor problem is NP-complete. If H is part of the input, then the Induced Minor problem remains NP-complete even when H is restricted to be a disjoint union of paths (see Proposition 3.16) or a cycle. In fact, both C_k-Induced Minor and P_k-Induced Minor are W[1]-complete when parameterized by k (see [26]).

However, they can both be solved in polynomial time when the length of the desired path or cycle is fixed: finding a path of length k as an induced minor is equivalent to finding the same graph as a induced subgraph, whereas the problem of finding cycles of length at least k was proved to solvable in polynomial time for every fixed k by Nikolopoulos and Palios [118]. The question whether the H-Induced Minor problem can be solved in polynomial time for every fixed tree was first asked at the AMS-IMS-SIAM Joint Summer Research Conference on Graph Minors in 1991, as mentioned in [57]. In the same paper, Fiala, Kamiński and Paulusma proved that this is true in the case where H is any fixed star that may be subdivided or any fixed double star, one side of which contains exactly 2 leaves. This allowed them to settle the computational complexity of H-Induced Minor for every fixed forest on at most 7 vertices, except one case, which is the graph depicted in Figure 3.8 below. Finally, the case where H is isomorphic to the disjoint union of two cycles of length 3 is also open (see [57]).

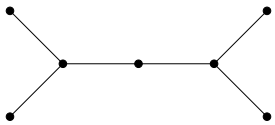


Figure 3.8: The only forest on at most 7 vertices for which the H-Induced Minor problem is not known to be solvable in polynomial time.

Note that the problem K_t-Induced Minor is FPT when parameterized by t. This follows from the fact that a graph contains K_t as an induced minor if and only if it contains it as a minor. Together with the result from [94] stating that H-Minor can be solved in quadratic time for every fixed H, it follows that K_t-Induced Minor can also be solved in quadratic time, for every fixed t. Note that the problem is NP-complete when t is part of the input, due to a result by Eppstein [52].

Contraction: Brouwer and Veldman proved in [23] that the problem H-Contractibility is NP-complete even when H is isomorphic to a path or a cycle of length 4, thereby already limiting greatly the potential classes of graphs H for which H-Contractibility could be solvable in polynomial time. However, they also showed that for every connected graph H on at most 4 vertices other than P_4 or C_4, the H-Contractibility problem is solvable in polynomial time. This was later extended by Levin, Paulusma and Woeginger in [106, 107], where they gave a complete classification for all graphs H on at most 5 vertices. They observed in particular that

the problem is NP-complete if H does not have a universal vertex, and polynomial-time solvable otherwise. However, it was later shown that this does not extend in general, when van 't Hof, Kamiński, Paulusma, Szeider and Thilikos [146] showed that there exists a graph H that has a universal vertex and for which H-Contractibility is NP-complete. It is still unknown if the problem is NP-complete for every graph H that does not have a universal vertex. Finally, note that K_t-Contractibility can be solved in quadratic time for every fixed t for the same reason as H-Induced Minor and therefore the problem is also NP-complete when t is part of the input.

3.7.2 Restrictions on G

In Section 3.4.1, we saw that the Minor, Induced Minor, Topological Minor, Induced Topological Minor, Subgraph Isomorphism and Induced Subgraph Isomorphism problems remain NP-complete even when G is restricted to be a planar subcubic graph. In [113], Matoušek and Thomas studied the computational complexity of these problems, as well as Contractibility and Dissolution, on graphs of bounded treewidth. They proved in particular the following:

Theorem 3.31 (Theorems 4.1 and 4.2 of [113])**.** *The* Minor*,* Induced Minor *and* Contractibility *problems are* NP*-complete on trees, even with one of the following restrictions on G and H:*

1. *G and H have bounded diameter,*
2. *G and H both have at most one of degree more than 5.*

Note that Theorem 3.31 is used in [10] (see Chapter 6) to prove that Contractibility, Induced Minor and Induced Subgraph Isomorphism are also NP-complete on trivially perfect graphs.

We now provide more detailed lists of results for several containment relations. Although the complexity of the problems related to induced minor, induced topological and contraction are mostly the same, we choose to provide a separate list for relation in order to ease readability. Moreover, even though these problems appear to have a similar complexity on many graph classes, the results often come from different sources.

Induced Minor: In [10], Belmonte, Heggernes and van 't Hof proved that the Induced Minor problem can be solved in polynomial time

when G is trivially perfect and H is threshold (see Chapter 6). Additionally, Golovach, Kamiński, Paulusma and Thilikos proved in [71] that the problem is NP-complete when the input graphs are split graphs. It is interesting to notice that the proof of Theorem 7 in [10] can easily be adapted to obtain that Induced Minor is NP-complete when G is split and H is threshold. Belmonte, Heggernes and van 't Hof [10] also proved that the problem remains NP-complete when both G and H are trivially perfect. Fiala, Kamiński and Paulusma recently studied the complexity of the problem in subclasses of claw-free graphs and showed that Induced Minor remains NP-complete when G and H are both line graphs [58].

In [72], Golovach, Kamiński, Paulusma and Thilikos proved that H-Induced Minor is W[1]-hard when G and H are split. However, it was proved independently in [10] and [72] that the problem is solvable in polynomial time for every fixed graph H when both G and H are split graphs. This result was then extended by Belmonte, Golovach, Heggernes, van 't Hof, Kamiński and Paulusma to the case where both G and H are chordal graphs in [7] (see also Chapter 7). Golovach, Kratsch and Paulusma [73] also proved recently that H-Induced Minor is W[1]-hard on co-bipartite graphs, but can be solved in polynomial time for every fixed graph H on AT-free graphs. Fellows, Kratochvíl, Middendorf and Pfeiffer proved in [55] that H-Induced Minor becomes FPT when G and H are planar graphs. This was later extended by van 't Hof, Kamiński, Paulusma, Szeider and Thilikos in [146] to the case where G is H^*-minor free for some fixed graph H^* and H is planar. However, it remains open to decide if the problem can be solved in polynomial time for every fixed H when G has bounded genus (see [55]). Finally, as we saw in Section 3.6, the problem H-Induced Minor can be expressed in monadic second-order logic, without using quantification over sets of edges, which implies that the problem can be solved in linear time on graphs of bounded clique-width, e.g., trivially perfect graphs, distance-hereditary graphs and graphs of bounded treewidth.

Induced Topological Minor The computational complexity of the Induced Topological Minor problem is somewhat similar to that of Induced Minor: the problem was also observed to be NP-complete on line graphs in [58] and split graphs [72]. Moreover, we can observe that Theorem 1 of [10] implies that for any two trivially perfect graphs G and H, G contains H as a topological minor if and only if G contains H as an induced subgraph. This observation, combined with the results of [10] regarding the complexity of Induced Minor on trivially perfect graphs,

implies that Induced Topological Minor can be solved in polynomial time when G is a trivially perfect graph and H is a threshold graph, but that it becomes NP-complete when the restriction on H is removed.

The H-Induced Topological Minor problem was showed in [72] to be W[1]-hard when G and H are split, but it can be solved in polynomial time for every fixed H. This easily generalizes to the class of P_l-free graphs, since the k-Induced Disjoint Paths problem can be solved in polynomial time for every fixed k, l. This follows from the fact that any induced path contains at most $k-1$ vertices, and therefore any solution contains at most $l \cdot (k-1)$ vertices. The result was also extended in [7] (see Chapter 7), where it was shown to be solvable in polynomial time for every fixed graph H when G is a chordal graph. Moreover, since the k-Induced Disjoint Paths problem can be solved in polynomial time on AT-free graphs [75] and claw-free graphs [76], it follows that the problem is polynomial for every graph H when G belongs to one of these two classes. Similarly, the problem is solvable in polynomial time when G and H are both planar due to the algorithm for the k-Induced Disjoint Paths problem of Kawarabayashi and Kobayashi [93]. Moreover, since they claimed in [91] that their algorithm can be extended to graphs of bounded genus, this would imply that H-Induced Topological Minor is polynomial when G and H both have bounded genus. Finally, as we saw in Section 3.6, the problem H-Induced Topological Minor can be expressed in monadic second-order logic, without using quantification over sets of edges, which implies that the problem can be solved in linear time on graphs of bounded clique-width.

Contractibility As for Induced Minor and Induced Topological Minor, the Contractibility problem can be solved in polynomial time when G is trivially perfect and H is threshold graphs, but remains NP-complete when both G and H are threshold [10]. Fiala, Kamiński and Paulusma also proved that the problem remains NP-complete when restricted to line graphs. Moreover, Golovach, Kamiński, Paulusma and Thilikos [71] showed that the problem is NP-complete when G and H are both split graphs, and Belmonte, Heggernes and van 't Hof [10] proved that this is the case even when H is restricted to be a threshold graph.

Regarding the parameterized complexity of the problem, Golovach, Kamiński, Paulusma and Thilikos proved that H-Contractibility is W[1]-hard when G and H are split [72]. This hardness result was proved to be tight independently in [10] and [72], where it was probed that H-Contractibility can be solved in polynomial time for every fixed graph

H when G is a split graph. This was first extended to the case where G is chordal and H is split or a tree [71], and later to the case where both G and H are only restricted to be chordal [7]. Golovach, Kratsch and Paulusma proved in [73] that the problem is also W[1]-hard on the class of co-bipartite graphs. However, it can be solved in polynomial time for every fixed triangle-free graphs H when G is an AT-free graph [73]. Very interestingly, Fiala, Kamiński and Paulusma proved in [58] that the problem P_7-Contractibility is NP-complete on line graphs. The parameterized complexity of the problem on graphs of bounded genus was first studied by Kamiński, Paulusma and Thilikos in [87], where it was shown that the problem can be solved in polynomial time for every fixed graph H. It was later proved to be FPT when G is planar and H is a contraction of a triangulated planar graph [88]. Furthermore, it was recently shown in [89] that the problem is FPT when both G and H have bounded genus. Finally, as we saw in Section 3.6, the problem H-Contractibility can be expressed in monadic second-order logic, without using quantification over sets of edges, which implies that the problem can be solved in linear time on graphs of bounded clique-width.

Subgraph and Induced Subgraph and Spanning Subgraph As we mentioned in previous sections, both H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism are W[1]-hard on general graphs, but can be solved in polynomial time for every fixed graph H. Moreover, they are NP-complete when H is part of the input. Damashke showed in [36] that Induced Subgraph Isomorphism remains NP-complete on the class of cographs. This result was later extended to trivially perfect graphs by Belmonte, Heggernes and van 't Hof [10], who also showed that this hardness result is in a sense tight by proving that the problem is polynomial when G is a trivially perfect graph and H is a threshold graph. In [96], Kijima, Otachi, Saitoh and Uno showed that Subgraph Isomorphism is polynomial on threshold graphs, chain graphs and cochain graphs. They also showed that Spanning Subgraph Isomorphism remains NP-complete on proper interval graphs, bipartite permutation graphs and trivially perfect graphs. Fiala, Kamiński and Paulusma also showed in [58] that Subgraph Isomorphism, Induced Subgraph Isomorphism and Spanning Subgraph Isomorphism all remain NP-complete on the class of line graphs.

Eppstein showed in [50] that both H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism can be solved in linear time on planar graphs. This was later extended to bounded genus in [51] and

then recently to graphs of bounded expansion by Nešetřil and Ossona de Mendez [116].

Marx and Schlotter [112] proved that H-Induced Subgraph Isomorphism is W[1]-hard on interval graphs, while Damashke observed in [36] that the problem remains NP-complete when both G and H are restricted to be disjoint unions of paths. Note that, as Damashke observed in [36], the Induced Subgraph Isomorphism problem is NP-complete on every class of graph that contains arbitrarily long induced paths, which include the class of H-free graphs whenever H is not a disjoint union of paths, and on the class of P_4-free graphs, i.e., cographs. Moreover, as again observed by Damashke [36], Induced Subgraph Isomorphism is easily seen to be solvable in polynomial time on P_3-free graphs. Note that Induced Subgraph Isomorphism can be solved in polynomial time on H-free graphs if and only if it can be solved in polynomial time on $\overline{H}$-free graphs, where $\overline{H}$ is the complement of H. This simply follows from the fact for any two graphs G and H, H is an induced subgraph of G if and only if $\overline{H}$ is an induced subgraph of $\overline{G}$. Hence we obtain a complete dichotomy for the complexity of Induced Subgraph Isomorphism on H-free graphs: the problem is polynomial if H is an induced subgraph of P_3 or $\overline{P_3}$, and it is NP-complete otherwise.

As observed in [7] (see Chapter 7), the H-Subgraph Isomorphism can be solved in linear time on chordal graphs, using the simple observation that if G contains a clique of size $|V(H)|$ then it contains H as a subgraph, and otherwise it has bounded treewidth (it is well-known that chordal graphs of bounded clique size have bounded treewidth); moreover, the largest clique of a chordal graph can be found in linear time. Finally, as with H-Induced Minor and H-Contractibility, both H-Subgraph Isomorphism and H-Induced Subgraph Isomorphism can be expressed in monadic second-order logic, without using quantification over sets of edges, which implies that the problem can be solved in linear time on graphs of bounded clique-width.

Chapter 4

Combinatorial aspects of containment relations in graphs

In the previous chapter, we brought our attention to computational aspects of containment relations in graphs. We now turn to more purely combinatorial aspects of containment relations. Even though the problems discussed in this section may appear first as being only of theoretical interest, we will see that they actually provide powerful tools to help solve problems of algorithmic nature.

4.1 Structure of graphs excluding a fixed pattern

Maybe the most natural question regarding containment relations is the following: what happens when we consider only graphs that do not contain a specific pattern? In other words, what structure do we gain from excluding a fixed graph with respect to some particular containment relation?

4.1.1 Minors

Wagner [148] proved that a graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor. Moreover, he precisely characterized the graphs that only exclude K_5 as a minor.

Theorem 4.1 (Wagner, 1937; see also [40])**.** *Let G be an edge-maximal K_5-minor free graph with $|V(G)| \geq 4$. G can be constructed recursively,*

by pasting along K_2s and K_3s, from plane triangulations and copies of the Wagner graph (see Figure 4.1).

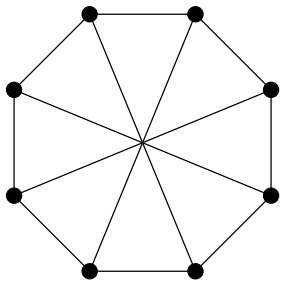


Figure 4.1: The Wagner graph.

The next step is naturally to ask what is the structure of graphs that do not contain a fixed clique as a minor. Robertson and Seymour gave such a description [130] by the means of decomposition as part of their Graph Minor project. Definitions 4.2, 4.3, 4.4 and 4.5 can be found in [130].

Definition 4.2 ($(\leq n)$-vertex extension)**.** *Let G and G_0 be two graphs such that G_0 is a subgraph of G. G_0 is an $(\leq n)$-vertex extension of G if it can be obtained from G by deleting at most n vertices.*

The set S of vertices of G such that $G - S = G_0$ is sometimes referred to as the set of *apices* of G. We now give the definition of an r-ring:

Definition 4.3 (r-ring)**.** *A graph G is an r-ring with perimeter $t_1, \ldots, t_n$ if $t_1, \ldots, t_n \in V(G)$ are distinct and there is a sequence $X_1, \ldots, X_n$ of subsets of $V(G)$ such that*

- *$X_1 \cup \ldots \cup X_n = V(G)$ and every edge of G has both ends in some X_i,*
- *$t_i \in X_i$ for $1 \leq i \leq n$,*
- *$X_i \cap X_k \subseteq X_j$ for $1 \leq i \leq j \leq k \leq n$,*
- *$|X_i| \leq r$ for $1 \leq i \leq n$.*

Informally, an r-ring is a graph of bounded pathwidth given with a path decomposition in which we choose one vertex t_i for each node X_i of the path decomposition such that t_i appears only in X_i.

Definition 4.4 (Outgrowth). *Let G_0 be a graph drawn in a surface Σ and let $\Delta_1, \ldots, \Delta_d \subseteq \Sigma$ be pairwise disjoint closed discs, each meeting the drawing only in vertices of G_0, and each containing no vertices of G_0 in its interior. For $1 \leq i \leq d$ let the vertices of G_0 in the boundary of (Δ_i) be $t_1, \ldots, t_n$ say, in order, and choose an r-ring G_i with perimeter $t_1, \ldots, t_n$, meeting G_0 just in $t_1, \ldots, t_n$ and disjoint from every other G_j; and let G be the union of $G_0, G_1, \ldots, G_d$. Such a graph G (and any graph isomorphic to it) is called an outgrowth by d r-rings of a graph in Σ.*

Informally, an outgrowth of a graph G_0 embedded in some surface Σ is obtained by "gluing" at most d r-rings to G_0 in such a way that each r-ring is drawn in a face of G_0. By combining the notions of $(\leq n)$-vertex extension and outgrowth by d r-rings, we obtain the definition of near embeddability in a surface:

Definition 4.5 (h-nearly embedded). *A graph G is said to be h-nearly embedded in a surface Σ if it is a $(\leq h)$-vertex extension of an outgrowth by at most h h-rings of a graph that can be embedded in Σ.*

The last definition we need before we can state Theorem 4.7 is that of h-clique-sum, which provides a way to "paste" graphs together along small separators:

Definition 4.6 (h-clique-sum). *Let G_1 and G_2 be two disjoint graphs, and h an integer. For $i \in \{1, 2\}$, let $W_i \subseteq V(G_i)$, be a clique of size h and let G'_i be the graph obtained from G_i by removing a set of edges (possibly empty) from the graph $G_i[W_i]$. Let $F : W_1 \to W_2$ be a bijection between W_1 and W_2. We define the h-clique-sum of G_1 and G_2 as the graph obtained by taking the union of G'_1 and G'_2, identifying w with $F(w)$ for every $w \in W_1$, and by removing all the multiple edges.*

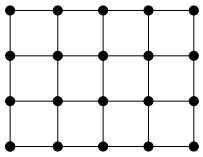
We are now ready to state Theorem 4.7, which is the main result of [130]:

Theorem 4.7 (Theorem 1.3 of [130]). *For every non-planar graph H, there exists an integer h, depending only on H, such that every graph excluding H as a minor can be obtained by h-clique-sums from graphs that can be h-nearly embedded in a surface Σ in which H cannot be embedded.*

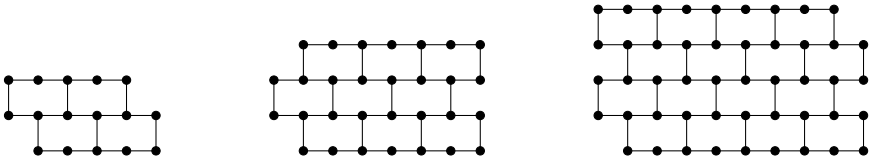
Recently, Kawarabayashi and Wollan [95] gave a much shorter proof of Theorem 4.7. Note that Theorem 4.7 requires H to be a non-planar graph. In the case where H is planar, there is no surface Σ in which H cannot be embedded. This condition is then replaced, as we will see

next, by the condition that G has bounded treewidth. This can easily be seen to be equivalent to replacing the h-nearly embeddable condition in Theorem 4.7 by "obtained by h-clique-sums from graphs that have at most h vertices".

Excluded minors and width parameters: Before we start discussing the relation between excluded minors and width parameters, we need to define two important classes of graphs that are tightly related to these two notions: grids and walls. The $n \times m$ *grid* is the graph with vertex set $\{u_{i,j} \mid 1 \leq i \leq n, 1 \leq j \neq m\}$ and edge set $\{u_{i,j}u_{i',j'} \mid |i'-i| + |j'-j| = 1\}$ (see Figure 4.2a for an illustration). Similarly, the elementary wall of height r is the graph with vertex set $\{u_{i,j} \mid 0 \leq i \leq 2r+1, 0 \leq j \leq r\} \setminus \{U_{0,0}, u_{2r+1,r}\}$ if r is even and $\{u_{i,j} \mid 0 \leq i \leq 2r+1, 0 \leq j \leq r\} \setminus \{u_{0,0}, u_{0,r}\}$ if r is odd, and such that there is an edge between any vertices $u_{i,j}$ and $u_{i',j'}$ if either $|i'-i| = 1$ and $j = j'$, or if $i = i', |j'-j| = 1$ and i and $\max\{j, j'\}$ have the same parity (see Figure 4.2b for an illustration). Note that both grids and walls are planar graphs, and walls are additionally subcubic.



(a) The 5×4 grid.



(b) The elementary walls of height $3, 4$ and 5.

Figure 4.2: Grids and walls.

It is a well-known fact that a graph has tree-width at most 1 if and only if it is a forest, i.e., K_3-minor free, and that it has treewidth at most 2 if and only if each of its biconnected components is a series parallel graph (see e.g. [18]). Moreover, it is easy to observe that if H is a non-planar graph, then the class of H-minor free graphs does not have bounded treewidth, since arbitrarily large grids are H-minor free (since taking minor preserves planarity) and large grids have large treewidth. Therefore, the class of graphs H for which H-minor free graphs have

bounded treewidth is a subclass of planar graphs. A natural question to ask is whether it is a strict subclass, i.e., does there exist a planar graph H^* such that H^*-minor free graphs do not have bounded treewidth? The answer to this question was proved to be negative by Robertson and Seymour in [126], where they proved that a graph has large treewidth if and only if it contains a large grid as a minor. Together with the fact that every planar graph is a minor of some sufficiently large grid, it implies Theorem 4.8.

Theorem 4.8 ([126, 134, 41, 92, 103])**.** *For every graph H, there is a constant w_H such that every graph that does not contain H as a minor has tree-width at most w_H if and only if H is planar.*

When Robertson and Seymour first proved Theorem 4.8 in [126], they did not provide an explicit value for the constant w_H. However, they mention in [134] that the value is huge, i.e., a tower of exponentials. In [134], Robertson, Seymour and Thomas provided for the first time an explicit value for w_H, i.e., $20^{O(|V(H)|^5)}$. Diestel et al. [41] later provided a shorter proof while obtaining a similar value for w_H. The upper bound on the value of w_H was later improved to $2^{O(|V(H)|^2 \cdot log|V(H)|)}$ by Kawarabayashi and Kobayashi [92] and recently again to $2^{O(|V(H)| \cdot log|V(H)|)}$ by Leaf and Seymour [103]. The best lower bound known for the value of w_H was given in [134], where they provided an $\Omega(|V(H)|^2 \cdot log|V(H)|)$ bound. It was conjectured that the value of w_H is polynomial, but only super-exponential upper bound were known until very recently, when Chekuri and Chuzhoy proved that w_H is upper bounded by $O(|V(H)|^{228})$ [25]. However, for some specific classes of planar graphs, polynomial upper bounds were already known. Examples include Birmelé, Bondy and Reed, who proved in [16] that excluding a prism as a minor forces polynomial treewidth. More recently, Raymond and Thilikos [123] gave low degree polynomial upper bounds for several families of planar graphs such as wheels and double wheels.

Similar results exist for graphs that exclude a fixed forest as minor and the graph parameter pathwidth. More precisely, Robertson and Seymour proved in [125] that a class of graphs has bounded pathwidth if and only if it excludes some fixed forest as a minor. They did not provide an explicit upper bound on the value of the pathwidth in [125], but Robertson, Seymour and Thomas mentioned in [15] that the constant obtained in [125] was huge, and improved it by showing that for every forest F, every graph with no minor isomorphic to F has path-width at most $|V(F)| - 2$.

4.1.2 Immersions

In the case of immersion, a decomposition theorem was very recently given independently by Wollan [152], and DeVos, McDonald, Mohar and Scheide [39]. We would like to note that compared to Theorem 4.7, both the statement and the proof of Theorem 4.11 are much simpler to understand. Before we state Theorem 4.11, we need the following definitions.

Definition 4.9 ((α,β)-bounded degree)**.** *Let G be a graph and $\alpha,\beta \geq 0$ positive integers. Then G has (α,β)-bounded degree if there exist at most α vertices of degree at least β.*

Definition 4.10 (Edge-sum)**.** *Let G, G_1, and G_2 be graphs and $k \geq 1$ an integer. The graph G is a k-edge sum of G_1 and G_2 if the following holds. There exist vertices $v_i \in V(G_i)$ such that $d_{G_i}(v_i) = k$ for $i \in \{1,2\}$, and a bijection $\pi : N_{G_1}(v_1) \to N_{G_2}(v_2)$ such that G is obtained from $(G_1 - v_1) \cup (G_2 - v_2)$ by adding an edge from $x \in V(G_1) \setminus \{v_1\}$ to $y \in V(G_2) \setminus \{v_2\}$ for every pair e_1, e_2 of edges such that e_i is incident to v_i for $i \in \{1,2\}$, the ends of e_1 are x and v_1, the ends of e_2 are y and v_2, and $e_2 = \pi(e_1)$. The edge sum is grounded if there exist vertices w_1 and w_2 in G_1 and G_2, respectively, such that for $i \in \{1,2\}$, $v_i \neq w_i$ and there exist k edge-disjoint paths linking v_i and w_i. If G can be obtained by a k-edge sum of G_1 and G_2, we write $G = G_1 \hat{\oplus}_k G_2$.*

Theorem 4.11 (Theorem 4 of [152])**.** *Let $t \geq 1$ be a positive integer. If G is a graph which does not admit K_t as a weak immersion, then either G has (t, t^2)-bounded degree or there exist graphs G_1, G_2 which do not have an immersion of K_t, and an integer $k < t^2$ such that G is given by a grounded edge sum $G_1 \hat{\oplus}_k G_2$. Moreover, $|V(G_1)|, |V(G_2)| < |V(G)|$.*

Note that Theorem 4.8, combined with Proposition 3.11 and the fact that if a graph H is a topological minor of a graph G, then H is also an immersion of G, implies that if a graph H is planar and subcubic, then the class of H-immersion free graphs has bounded treewidth. Note that this sufficient condition is also necessary, since every graph that immerses in a graph G isomorphic to an arbitrarily large wall is planar subcubic. These two facts can then be combined as follows, as a corollary of Theorem 4.8.

Corollary 4.12. *For every graph H, there is a constant w_H such that every graph that does not contain H as an immersion has tree-width at most w_H if and only if H is planar and subcubic.*

4.1.3 Topological minors

Finally, a decomposition theorem for the graphs that do not contain some fixed graph as a topological minor was recently given by Grohe and Marx [80].

Theorem 4.13 (Corollary 4.4 of [80]). *For every graph H, there exist constants h, t such that every graph that does not contain H as a topological minor may be constructed by clique-sums, starting from graphs G_0 that satisfies one of the following properties:*

- *G_0 has (t, t)-bounded degree,*
- *G_0 is h-nearly embedded in a surface of genus h.*

This result was later strengthened by Dvořák [49], who showed that the second condition can be replaced by: G_0 is h-nearly embedded in a surface in which H cannot be embedded (similarly to Theorem 4.7). It is most interesting to observe that the decomposition theorem of Marx and Grohe for H-topological minor free graphs appears to be a perfect combination of the decomposition theorems for H-minor free graphs and H-immersion free graphs. Indeed, Theorem 4.13 above can be rewritten in the following way:

Corollary 4.14 (Theorem 4.1 of [80]). *For every graph H, there exist constants p, q such that every graph that does not contain H as a topological minor may be constructed by clique-sums, starting from graphs G_0 that satisfy one of the following properties:*

- *G_0 does not contain K_p as a minor,*
- *G_0 does not contain K_q as an immersion.*

It is interesting that excluding a graph as a topological minor seems to amount to forbidding it as either a minor or an immersion.

Note also that, while a general decomposition theorem is now known, it is still interesting to study the structure of H-topological minor free graphs for some specific graph H. For instance, Farr studied the structure of W_4 and W_5-topological minor free graphs in [53]. Robinson and Farr [136] later provided a decomposition theorem for W_6-topological minor free graphs and W_7-topological minor free graphs [135]. A fundamental open problem is to provide a precise characterization of graphs without K_5-topological minor, similar to Wagner's theorem (Theorem 4.1) for graphs that exclude K_5 as a minor.

Additionally, note that the relation between graphs of small treewidth and graphs that exclude a fixed graph as a topological minor is identical to the relation between graphs of small treewidth and graphs that exclude a fixed graph as an immersion. This follows again from the combination of Theorem 4.8, Proposition 3.11 and the fact that large walls are planar subcubic graphs and have large treewidth.

Corollary 4.15. *For every graph H, there is a constant w_H such that every graph that does not contain H as a topological minor has tree-width at most w_H if and only if H is planar and subcubic.*

4.1.4 Structure of H-free graphs for some graphs H

While general decomposition theorems describing the structure of graphs not containing a fixed graph as a pattern are known for the minor, topological minor and immersion relations, this fails to be the case for other containment relations, in particular induced subgraph. Many fundamental and well-studied classes of graphs are defined in terms of some forbidden induced subgraph. Maybe one of the most famous class of H-free graphs is the class of claw-free graphs. The structure of claw-free graphs was studied by Chudnovsky and Seymour, who provided a decomposition theorem revealing their structure [30]. This powerful decomposition theorem was later used to devise efficient algorithms for various problems (see e.g. [84]). Chudnovsky also recently studied the structure of bull-free graphs [27], where the bull is the graph depicted on Figure 4.3, and provided a characterization for the graphs that belong to this class.

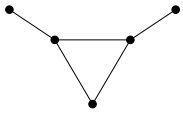


Figure 4.3: The bull graph.

Another important family of graphs that have received much attention is the class of P_k-free graphs, for some integer $k \geq 1$. Observe first that the class of P_1-free graphs contains only the empty graph, and the class of P_2-free graphs consists of all edgeless graphs. It is well-known that the graphs that do not contain P_3 as an induced subgraph are exactly those formed by disjoint union of cliques, also known as *clusters*. The first non-trivial case is the class of P_4-free graphs, which is well-known to be equivalent to the class of cographs (see e.g. [78, 22]). The class of P_5-free graph was studied by Bacsó and Tuza in [5], where they proved that every

connected P_5-free graph has either a dominating clique, or a dominating induced P_3. Moreover, they proved that a connected graph is $\{P_5, C_5\}$-free if and only if each connected induced subgraph of it contains a dominating clique. This result was extended by Liu, Peng and Zhao [108], who proved that a graph is P_5-free if and only if each connected induced subgraph of it contains a dominating induced C_5 or a dominating clique. They also provided a necessary and sufficient condition for a graph to be P_6-free, which was later improved by van 't Hof and Paulusma [147] who proved that a graph is P_6-free if and only if each connected induced subgraph of it contains a dominating induced C_6 or a dominating complete (not necessarily induced) bipartite subgraph. For the general case, a necessary and sufficient condition for a graph to be P_k-free was given by Bacsó and Tuza [4].

We would also like to briefly mention that in some cases, more specific properties of H-free graphs are studied, as for example χ-boundedness: a class of graph $\mathcal{G}$ is χ-bounded if there exists a function f such that for every graph $G \in \mathcal{G}$, the chromatic number of G is at most $f(\omega(G))$, where $\omega(G)$ is the size of the largest clique in G. Gyárfás and Sumner conjectured that T-free graphs are χ-bounded for every tree T (see [138]). Scott [138] proved that the conjecture holds when trees are forbidden as induced topological minor, and he further conjectured that H-induced topological minor free graphs are χ-bounded for every fixed graph H. Chudnovsky, Penev, Scott and Trotignon proved that H-induced topological minor free graphs are χ-bounded when $H \in \{bull, paw, necklace\}$ [29], where the paw is the graph depicted on Figure 4.4 and a necklace is a graph obtained from a path by choosing a matching such that no edge of the matching is incident with an endpoint of the path, and for each edge of the matching, adding a vertex adjacent to the ends of this edge.

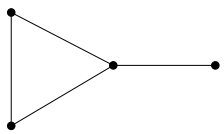


Figure 4.4: The paw graph.

Finally, note that the question of determining the chromatic number of K_t-minor free (or equivalently K_t-induced minor free or K_t-contraction free) graphs is closely related to the famous Hadwiger's Conjecture, which states that K_t-minor free graphs have chromatic number at most $k-1$ (see e.g. [133] for the case $k = 6$).

4.2 Well-quasi-orders

An important notion in the study of containment relations is that of well-quasi-orders. A reflexive and transitive binary relation $\leq$ over a set S is called a *quasi-order*. Note that partial-orders are a special case of quasi-orders. The pair $(S, \leq)$ forms a *well-quasi-order* if, for every infinite sequence of elements $x_1, x_2, x_3, \ldots \subseteq S$, there exist indices $1 \leq i \leq j$ such that $x_i < x_j$. It is also well-known that $(S, \leq)$ forms a well-quasi-order if and only if the two following conditions are satisfied:

(i) It is well-founded, i.e., there is no infinite sequence of distinct elements $x_1, x_2, x_3, \ldots \subseteq S$ such that $x_1 \geq x_2 \geq x_3 \geq \ldots$.

(ii) There is no infinite antichain, where an antichain is a set of pairwise non-comparable elements of S.

In such a case, we say that S is well-quasi-ordered under $\leq$. The following well-known property of well-quasi-orders plays a fundamental role in applications to algorithmic graph theory:

Observation 4.16. *Let $\mathcal{G}$ be a class of graphs and $\leq$ a containment relation. The following are equivalent:*

- *$\mathcal{G}$ is well-quasi-ordered under $\leq$.*
- *Every class $\mathcal{G}' \subseteq \mathcal{G}$ of graphs can be defined by a finite set of graphs forbidden under $\leq$. In other words, there exists a finite set of graphs $\mathcal{F}_{\mathcal{G}}$ such that for every graph G, it holds that $G \in \mathcal{G}$ if and only if there is no graph $F \in \mathcal{F}_{\mathcal{G}}$ such that $F \leq G$.*

Note that each of the 14 containment relations defined in Chapter 3 clearly defines a well-founded partial-order. Therefore, to answer the question whether the set of all graphs is well-quasi-ordered under each of them, it remains to decide whether there exist infinite antichains for each of these relations. Kruskal's theorem [100] is certainly one of the most fundamental results in the theory of well-quasi-orders of graphs. This theorem states that trees are well-quasi-ordered by the topological minor relation, even when the vertices are labeled with elements from a well-quasi-order. Note that Kruskal's theorem implies in particular that trees are well-quasi-ordered under the minor relation. This was later extended to graphs of bounded treewidth in [127] and graphs of bounded genus in [128] by Robertson and Seymour in the Graph Minors series, whose goal was to prove the so-called Wagner's Conjecture, which states that the set of all

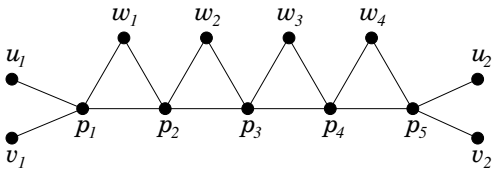
finite graphs is well-quasi-ordered under the minor relation. This conjecture was finally proved to be true at the culminating point of the Graph Minor series [131], and is now commonly referred to as the Graph Minor Theorem. Parallel to Wagner's Conjecture for minors, Nash-Williams conjectured in [114] that graphs are well-quasi-ordered under the immersion relation. He later conjectured that this holds even when considering strong immersion. Robertson and Seymour proved that Nash-Williams' Conjecture holds for weak immersion in [132]. They also mentioned that the the conjecture for strong immersion appears to be true as well, but that the proof is far more complex: "It seemed to us at one time that we had a proof of the stronger, but even if it was correct it was very much more complicated, and it is unlikely that we will write it down." [132].

The next question to ask is of course under which other relations the set of all graphs is well-quasi-ordered. Before answering this question, let us first make the following simple observation.

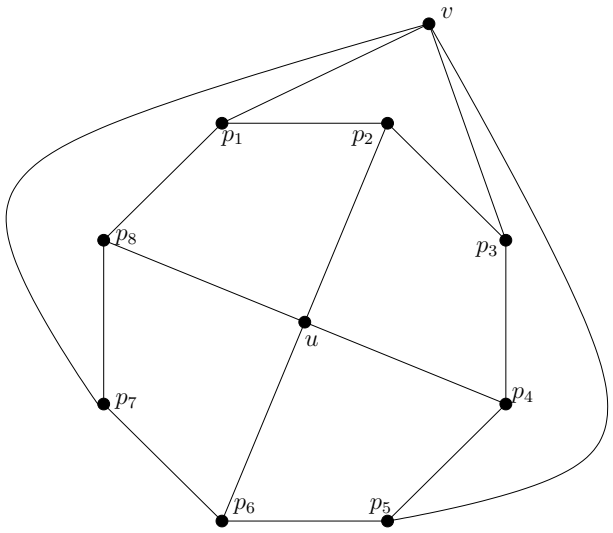
Observation 4.17. *Let $\leq_1, \leq_2$ be two partial orders on graphs, such that for any two graphs G and H, if $H \leq_1 G$, then $H \leq_2 G$. Then, for any class of graphs $\mathcal{G}$ closed under $\leq_2$, if $\mathcal{G}$ is well-quasi-ordered under $\leq_1$, then it is well-quasi-ordered under $\leq_2$ as well.*

Observation 4.17 simply follows from the fact that any antichain for the $\leq_2$ partial order is an antichain for $\leq_1$ as well. The main consequence of Observation 4.17 is that, when considering the hierarchy of relations depicted in Figure 3.5, if there is an arrow from relation $\leq_1$ to relation $\leq_2$, then any class of graphs closed under $\leq_2$ and well-quasi-ordered under $\leq_1$ is well-quasi-ordered under $\leq_2$ as well. For example, since there is an arrow from the topological minor relation to the minor relation in Figure 3.5, then every minor-closed class of graphs that is well-quasi-ordered under topological minor is well-quasi-ordered under minor as well. This means, informally, that more "restricted" relations are more general from the point of view of well-quasi-orders. With this observation, it should come as no surprise that the topological minor and induced minor relations (and hence all the relations located above them in Figure 3.5) do not form well-quasi-orders for the class of all graphs. A well-known antichain for the topological minor relation is the class of graphs $\{G_i \mid i \geq 1\}$, where G_i is the graph with vertex set $\{p_1, \ldots, p_{i+1}\} \cup \{w_1, \ldots, w_i\} \cup \{u_1, v_1, u_2, v_2\}$ and edge set $\{p_j p_{j+1} \mid 1 \leq j \leq i\} \cup \{w_j p_j, w_j p_{j+1} \mid 1 \leq j \leq i\} \cup \{u_1 p_1, v_1 p_1, u_2 p_{i+1}, v_2 p_{i+1}\}$. The graph G_4 is depicted on Figure 4.5a. Concerning the induced minor relation, a well-known antichain is the class of graphs $\{H_i \mid i \geq 1\}$, where H_i is the graph with vertex set $\{p_1, \ldots, p_{2i}\} \cup$

$\{u, v\}$ and edge set $\{p_j p_{j+1} \mid 1 \leq j \leq 2i-1\} \cup \{p_1 p_{2i}\} \cup \{up_{2j}, vp_{2j-1} \mid 1 \leq j \leq i\}$. The graph H_4 is depicted as shown on Figure 4.5b.



(a) The graph G_4 of the antichain for the topological minor relation.

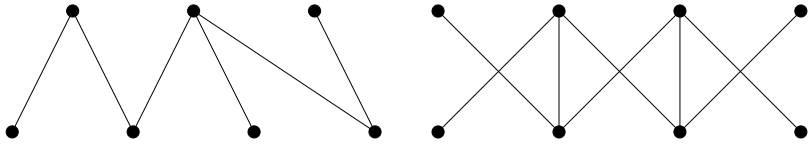


(b) The graph H_4 of the antichain for the topological minor relation.

Figure 4.5: Infinite antichains for the topological minor and induced minor relations.

However, it is still possible to obtain well-quasi-orders for some of these relations if we restrict them to particular classes of graphs. Kruskal's theorem, for instance, can be generalized to the class of subcubic graphs by combining the fact that graphs are well-quasi-ordered under minors with Proposition 3.11, which states that the minor and topological minor relations are equivalent on subcubic graphs. In [144], Thomas proved that series-parallel graphs are well-quasi-ordered under induced minor, while Ding [43] proved that interval graphs are not well-quasi-ordered under induced minor, but chordal graphs of bounded clique size are. Recently, Fellows, Hermelin and Rosamond studied well-quasi-orders on sub-

classes of graphs of bounded treewidth [54]. They proved that graphs with bounded vertex cover number are well-quasi-ordered under induced subgraphs, graphs of bounded feedback vertex set number under topological minors, and graphs of bounded circumference under induced minors. Damashke [37] showed that cographs are well-quasi-ordered under induced subgraphs. In fact, he even proved the stronger statement that the class of H-free graphs is well-quasi-ordered under induced subgraphs if and only H is an induced subgraph of P_4. In [42], Ding studied well-quasi-orders of subclasses of bipartite graphs and showed that the class of $\{P_7, J_1, J_2\}$-free bipartite graph is well-quasi ordered under induced subgraphs, where J_1 and J_2 are the graphs depicted in Figure 4.6. He also asked whether his result could be extended to the class of P_7-free bipartite graphs. This question was answered by Korpelainen and Lozin [99], who proved that P_7-free bipartite graphs are not well-quasi-ordered under induced subgraphs, but that P_6-free bipartite graphs are.



(a) J_1. (b) J_2.

Figure 4.6: Graphs J_1 (left) and J_2 (right).

Considering the fact that all graphs are not well-quasi-ordered under some relations, e.g., subgraph, a natural question arises: is there a particular set of graphs that every antichain has to contain? More precisely, given a containment relation $\leq$ under which the set of all graphs is not well-quasi-ordered, does there exist an infinite antichain $\mathcal{A}$ such that for every family of graphs $\mathcal{F}$ closed under $\leq$, $\mathcal{F}$ is well-quasi-ordered under $\leq$ if and only if $\mathcal{F} \cap \mathcal{A}$ is finite? Ding first defined this notion in [44], where he calls such an antichain $\mathcal{A}$ a *canonical antichain*. Actually, a canonical antichain for the subgraph relation had already been discovered by Ding earlier in [42], where it was proved that $\{C_i \mid i \geq 3\} \cup \{F_i \mid i \geq 1\}$ is a canonical antichain under the subgraph relation, where F_i is obtained from P_i, with endpoints p_1 and p_i, by adding vertices u_1, u_2 and v_1, v_2 that are made adjacent to p_1 and p_i respectively (see Figure 4.7). Moreover, Ding proved in [44] that there is no canonical antichain for the induced subgraph relation.

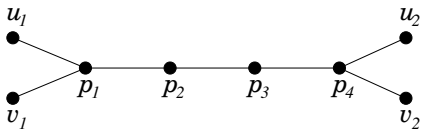


Figure 4.7: Graph F_4.

More recently, Lozin and Mayhill [111] studied canonical antichains for the classes of proper interval graphs and bipartite permutation graphs, and they proved that both classes, although known not to be well-quasi-ordered under the induced subgraph relation, admit canonical antichains.

We conclude this section by discussing some algorithmic consequences of well-quasi-orders. The following well-known observation constitutes one of the most important consequences of well-quasi-orders in algorithmic graph theory. Consider for instance the minor relation and recall that

(i) H-Minor can be solved in quadratic time for every input graph G and fixed graph H, and

(ii) the set of all graphs is well-quasi-ordered under the minor relation.

Recall that, as seen in Observation 4.16, (ii) implies that every class of graphs $\mathcal{G}$ closed under minor can be defined by a finite set $\mathcal{F}$ of forbidden minors. Therefore, given a graph G as input, we can use the algorithm for H-Minor from [94] and test if F is a minor of G for every graph $F \in \mathcal{F}$. If there is a graph $F \in \mathcal{F}$ that is a minor of G, then we conclude that $G \notin \mathcal{G}$; otherwise we conclude that $G \in \mathcal{G}$. Since the set $\mathcal{F}$ is finite, this procedure allows us to decide if G belongs to the class $\mathcal{G}$ when we are given the finite obstruction set $\mathcal{F}$ of $\mathcal{G}$. This particularly means that there is a quadratic-time algorithm for testing membership in any minor-closed graph class $\mathcal{G}$. Moreover, a similar argument can be applied to other containment relations $\leq$ and classes of graphs $\mathcal{G}$ such that $\mathcal{G}$ is well-quasi-ordered under $\leq$ and there exists an efficient algorithm to test whether $H \leq G$ for every $G \in \mathcal{G}$ and fixed H.

4.3 Obstructions

In the previous section, we have discussed several results regarding well-quasi-orders for graphs under various relations. As we saw in Observation 4.16, well-quasi-orders imply the existence of finite obstruction sets, and these obstruction sets can be used to derive efficient algorithms.

Therefore it is of primary importance to determine such obstruction sets and to compute them, whenever possible. The first question that comes to mind is whether this can always be done. This question was answered negatively by Fellows and Langston for the minor relation [56].

Theorem 4.18 ([56]). *There is no algorithm to compute, from a finite description of a minor-closed family $\mathcal{F}$ of graphs as represented by a Turing machine that accepts precisely the graphs in $\mathcal{F}$, the set of obstructions for $\mathcal{F}$.*

In addition, as mentioned in [56], the proof of Theorem 4.18 can easily be modified to handle other well-quasi-orders. Despite this uncomputability result, small obstruction sets together with efficient algorithms to decide containment relations are powerful tools to design efficient algorithms to test various graph properties. This might explain the wealth of results that have been obtained regarding the characterization of obstruction sets for various classes of graphs. In the remainder of this section, we provide a brief overview of these results.

First, we would like to mention two general results about the computability of obstruction sets for the minor and immersion relation respectively. The first of these two results is due to Adler, Grohe and Kreutzer [1].

Theorem 4.19 ([1]). *There is an algorithm that, given excluded minor characterizations of two classes of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ closed under minor, computes such a characterization for $\mathcal{G}_1 \cup \mathcal{G}_2$.*

A similar result was recently proved by Giannopoulou, Salem and Zoros [68] for immersions.

Theorem 4.20 ([68]). *There is an algorithm that, given excluded immersion characterizations of two classes of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ closed under immersion, computes such a characterization for $\mathcal{G}_1 \cup \mathcal{G}_2$.*

Additionally, we would like to mention another general result proved in [1]. An *apex graph* G over a class of graphs $\mathcal{G}$ is a graph from which one vertex can be removed to obtain a graph in $\mathcal{G}$.

Theorem 4.21 ([1]). *There is an algorithm that, given an excluded minor characterization of class of graphs $\mathcal{G}$, computes an excluded minor characterization for the class of all apex graphs over $\mathcal{G}$.*

We now give an overview of specific results about the identification of obstruction sets for various classes of graphs. First, we consider the classes

defined in Section 2.2. The following characterizations are well-known and easy to deduce:

- A graph is a forest if and only if it does not contain K_3 as a minor, or equivalently as a topological minor;
- A graph is chordal if and only if it does not contain C_4 as a topological minor.

Moreover, it has been shown that:

- A graph is an interval graph if and only if does not contain any of the graphs in Figure 4.8 as an induced minor [104].

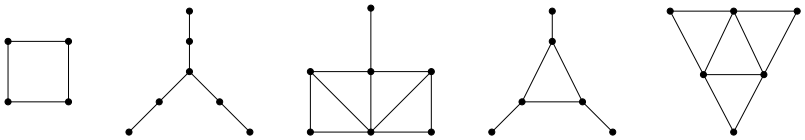


Figure 4.8: Induced minor obstructions for interval graphs.

- A graph is proper interval if and only if it is claw-free and interval, or equivalently, it does not contain any of the graphs in Figure 4.9 as an induced subgraph [149].

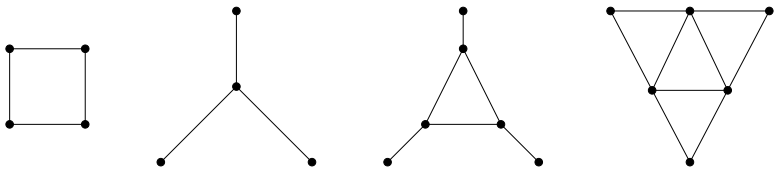


Figure 4.9: Induced subgraph obstructions for proper interval graphs.

- A graph is split if and only if does not contain $2K_2, C_4$ or C_5 as an induced subgraph [60].
- A graph is a threshold graph if and only if does not contain $2K_2, C_4$ or P_4 as an induced subgraph [31].

It is easy to observe that a simple graph has degree at most d if and only if it does not contain $K_{1,d}$ as a topological minor, or equivalently as an immersion. This holds even on multigraphs for topological minor, however for immersions one needs to forbid all multigraphs having exactly

d edges and whose underlying simple graph is isomorphic to $K_{1,d'}$, with $1 \leq d' \leq d$.

The famous theorems of Kuratowski [101] and Wagner [148] state that a graph is planar if and only if it contains neither K_5 nor $K_{3,3}$ as topological minor or as minor respectively. Similarly, a graph has genus at most g if it can be embedded in a surface of Euler genus g without crossing edges. Planar graphs are exactly the graphs of genus 0. Moreover, Robertson and Seymour proved in [128] that Wagner's theorem generalizes to arbitrary surfaces, in the sense that for any integer g, there is a finite family of graphs $\mathcal{F}_g$ such that a graph has genus at most g if and only if it does not contain any of the graphs in $\mathcal{F}_g$ as a minor. In particular, the set of minor obstructions for graphs that can be embedded in the projective plane was given by Glover and Huneke in [69], where they showed that there are 35 such obstructions. Moreover, it was proved by Neufeld [115] that the set of obstructions for graphs that can be embedded in torus contains at least 900 graphs.

Another significant amount of research regarding obstructions for various relations concerns so-called width parameters, for which obstruction sets are determined when the value is small. The parameter treewidth undoubtedly stands as one of the most well-studied width parameters, and it is therefore not surprising that obstruction sets for graphs of small treewidth are well-known. In particular, a graph has treewidth at most 1 if and only it does not contain K_3 as a minor, i.e., if it is a forest. It was also proved by Arnborg, Proskurowski and Corneil [3] that a graph has treewidth at most 2 if and only if it does not contain K_4 as a minor, or equivalently if its biconnected components are series-parallel. In the same paper, they proved that a graph has treewidth at most 3 if and only if it does not contain any of the graphs depicted in Figure 4.10 [3].

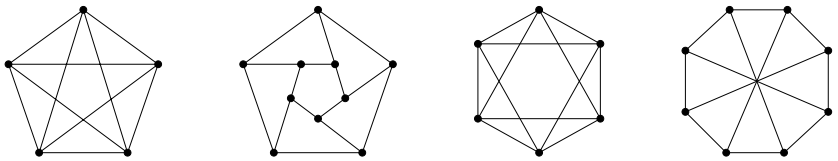


Figure 4.10: Minor obstructions for treewidth at most 3.

Pathwidth is a graph parameter closely related to treewidth and for which obstructions sets have also been identified. It is well-known that graphs of pathwidth at most 1 are exactly caterpillars, which are exactly the graphs that contain neither K_3 nor the graphs obtained from the claw by subdividing every edge exactly once as a minor. Kinnersley and

Langston [98] determined the minor obstruction set for graphs of pathwidth at most 2, which contains 110 elements, while it has be shown by Kinnersley that the obstruction set of graphs of pathwidth at most 3 contains at least at least 122 million elements [97]. In [14], Belmonte, van 't Hof, Kamiński, Paulsma and Thilikos studied the structure of graphs of small carving-width and proved that graphs of carving-width at most 3 are exactly graphs of treewidth at most 2 and maximum degree at most 3. This allowed them to obtain that immersion obstructions for graphs of carving-width at most 3 are exactly the graphs shown in Figure 4.11.

Figure 4.11: Immersion obstructions for carving-width at most 3.

Another well-known graph parameter that has also been widely studied is the vertex cover number, i.e., the minimum number of vertices to be deleted from a graph to make it edgeless. Cattell and Dinneen gave in [24] all the obstructions for the minor relation for graphs of vertex cover number k, with k at most 5. Later, Dinneen and Xiong provided the minor obstructions for vertex cover number at most 6 [47], and recently Dinneen and Versteegen gave those for vertex cover number at most 7 [46]. For the feedback vertex set parameter, Cattell, Dinneen and Fellows gave in [45] all the minor obstructions for graphs having feedback vertex set number at most k, with $k \leq 2$. There are 2 such obstructions for $k = 1$ and 24 connected ones for $k = 2$, while the disconnected ones can be obtained by taking the disjoint union of obstructions for $k = 1$.

Chapter 5

Conclusion

With this chapter we conclude Part I of this thesis. Here we mention some additional operations and containment relations that have not been studied in this thesis. In addition, we list some questions that remain open and explain why their solution is important. Finally, we give a short overview of Part II of this thesis.

5.1 Other operations and containment relations

In Chapter 3, we briefly mentioned a few additional operations and containment relations that were not covered in Chapters 3 and 4. The operations mentioned there were edge addition and edge subdivision, and the vertex-minor relation was mentioned as well. Here we would like to discuss further these notions and several additional operations and relations that were not natural to mention in the main body of Part I. First, we define a few additional operations. Let $G = (V, E)$ be a graph.

Vertex addition: The operation of *adding* a vertex w to G yields a graph $G' = (V \cup \{w\}, E)$, such that $w \notin V$.

Vertex identification: The operation of *identifying* two non-adjacent vertices u, v of G yields a graph G' with vertex set $(V \setminus \{u, v\}) \cup \{w\}$, such that $w \notin V$, and edge set $(E \setminus \{xy \in E \mid x \in \{u, v\}, y \in V(G)\}) \cup \{wz \mid z \in N(u) \cup N(v)\}$. Equivalently, G' is obtained by adding the edge uv to G and then contracting it.

Let Δ denote the symmetric difference operation, i.e., given two sets X and Y, $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$.

Local complementation: The operation of applying *local complementation* at a vertex w of G, denoted $G*w$, yields a graph $G' = (V, E\Delta\{xy \mid w \in N(x) \cap N(y), x \neq y\})$.

Edge pivoting: Let uv be an edge of G and $V_1 = N(u) \cap N(v), V_2 = N(u) \setminus N(v)$ and $V_3 = N(v) \setminus N(u)$. The operation of *pivoting* uv, denoted $G \wedge uv$, yields a graph $G' = (V, E\Delta\{xy \in (V_1 \times V_2) \cup (V_2 \times V_3) \cup (V_3 \times V_1)\})$. Equivalently, $G' = G * u * v * u$.

Vertex elimination: The operation of *eliminating* a vertex w of G yields a graph $G' = (V \setminus \{w\}, (E \setminus \{xw \mid x \in V\}) \cup \{xy \mid x, y \in N(w)\})$.

We are now ready to define some new containment relations based on these additional operations.

5.1.1 Homomorphism

Let G, H be two graphs. G is said to be *homomorphic* to H, or equivalently *H-colorable*, if there is a mapping $f : V(G) \to V(H)$ such that, for any two vertices $u, v \in V(G)$, if $uv \in E(G)$ then $f(u)f(v) \in E(H)$. See the monograph by Hell and Nešetřil on graph homomorphism [83] for more details. If H does not have any self-loops, then G is homomorphic to H if and only if the latter can be obtained from the former by identifying vertices, adding vertices and adding edges. Maybe the most important result regarding the computational complexity of the H-Homomorphism problem is the dichotomy theorem of Hell and Nešetřil [82]:

Theorem 5.1 ([82])**.** *The problem H-Homomorphism can be solved in polynomial time for every input graph G if H is bipartite, and is* NP*-complete otherwise.*

In particular, it is interesting to note that a graph G is homomorphic to K_t if and only if G is t-colorable, for any positive integer t.

We would also like to mention that different variants of homomorphisms can be defined by requiring the mapping f to have special properties. Particular examples include the cases where f is required to be surjective, injective or bijective. Observe that the two latter cases are equivalent to subgraph isomorphism and spanning subgraph isomorphism respectively. There is a surjective homomorphism from G to H if and only if H can obtained from G by identifying vertices and adding edges, i.e., unlike with homomorphism, we are not allowed to add vertices. The Surjective Homomorphism problem has been studied, e.g., in [74].

5.1.2 Elimination

The elimination operation has been extensively studied in the context of Gaussian elimination and sparse matrix computations, since eliminating a vertex simulates in graphs the elimination of a variable from subsequent rows during Gaussian elimination of symmetric matrices [120]. The computational complexity of the ELIMINATION problem, i.e., the problem of deciding if a graph H can be obtained from a graph G by applying a sequence of vertex eliminations, was first studied by Golovach, Heggernes, van 't Hof, Manne, Paulusma and Pilipczuk in [70]. They proved that the H-ELIMINATION problem is W[1]-hard when parameterized by $|V(H)|$ and W[2]-hard when parameterized by $|V(G)| - |V(H)|$, even in the case where H is restricted to be a complete graph. Additionally, they show that the problem admits a vertex-kernel of size linear in $|V(H)|$ when G is connected and H is complete. Finally, they show that ELIMINATION is NP-complete when G is restricted to be a tree, but becomes polynomial if H is restricted to be a tree instead.

5.1.3 Vertex-minor and pivot-minor

We now consider the local complementation and edge pivoting operations. We would first like to note that the local complementation operation is the only operation we have seen until now which does not change the number of vertices or edges of the graph in a monotone way, since the number of vertices remains unchanged after applying a local complementation, and the number of edges can either increase or decrease. As an example, consider the fact that $K_{1,t}$ and K_{t+1} can be obtained from each other by applying one local complementation. It is also interesting to observe that for any graph G and vertex u of G, we have $G*u*u = G$, which implies that for any two graphs G and G', if G' can be obtained from G by a sequence $\mathcal{S}$ of local complementations, then G can be obtained from G' by applying a sequence $\mathcal{S}'$ of local complementations. In particular, we can take $\mathcal{S}'$ to be the same sequence as $\mathcal{S}$ in reverse order. Moreover, the graphs G and G' are said to be *locally equivalent*. As the name suggests, the set of graphs locally equivalent to a graph G forms indeed an equivalence class. Bouchet showed in [20] that there is an algorithm that decides in polynomial time whether two graphs are locally equivalent. He also proved in [19] that any two locally equivalent trees are isomorphic.

With the local complementation and edge pivoting operations at hand, we can define the vertex-minor and pivot-minor relations. A graph H is a vertex-minor of a graph G if H can be obtained from G by applying

local complementations and vertex deletions. Pivot minors are defined similarly by replacing the local complementation operation by edge pivoting. Vertex minors were studied in particular by Bouchet under the name *l-reductions*, and he provided the set of vertex-minor obstructions for the class of circles graph [21]. Geelen and Oum [67] recently complemented this result by providing the set of pivot-minor obstructions for circle graphs. Interestingly, the vertex-minor and pivot-minor operations have strong connections with the graph parameters branch-width, rank-width, and the minor operation defined for matroids. In particular, Oum proved in [86] that the branch-width of a binary matroid $\mathcal{M}$ is exactly one more than the rank-width of its fundamental graph. In the same paper, he proved that a fundamental graph[1] of a minor of a binary matroid $\mathcal{M}$ is a pivot-minor of a fundamental graph of $\mathcal{M}$.

5.1.4 Beyond graphs

In the previous section, we mentioned the notion of minor for the so-called matroids. We do not provide a definition of matroids here, as this is not within the scope of this work, however the interested reader may refer to the monograph by Oxley [119]. We simply want to briefly mention that, although we only discussed containment relations on graphs, the notion of containment can be applied to any combinatorial structure in essentially the same way. Two particular examples include directed graphs and matroids, to which several graph containment relations can be extended, as we saw with the minor relation for matroids. Obviously, the notions related to containment relations that we discussed in the previous chapters, such as obstructions, well-quasi-orders and computational complexity of the associated decision problems are also well-studied on these structures, see, e.g. [65, 66] for examples regarding matroids, and [28, 62, 122] for examples regarding directed graphs.

5.2 Some open problems

In this section we discuss some open question. Several have already been mentioned in previous chapters and some are long standing open problems. In Section 4.1, we saw that Wagner gave a precise characterization of K_5-

[1] A fundamental graph of a binary matroid $\mathcal{M}$ is a bipartite graph with a bipartition $(B, E(\mathcal{M}) \setminus B)$ such that B is a basis of $\mathcal{M}$, and $e \in B$ and $f \in E(\mathcal{M}) \setminus B$ are adjacent if and only if e is in the fundamental circuit of f with respect to B. Refer to the monograph by Oxley [119] for more details.

minor free graphs in [148]. The following question is among the biggest open problems in structural graph theory:

Question 1. What is the structure of K_5-topological minor free graphs?

Note that here "structure" can be understood in many ways, and any characterization of K_5-topological minor free graphs would be interesting. However, it would be particularly interesting to obtain a decomposition theorem similar to that of Wagner for K_5-minor free graphs. We now turn our attention towards questions of algorithmic nature.

Question 2. Can H-Induced Minor be solved in polynomial time for every fixed tree H?

We mentioned Question 2 already in Section 3.7, together with the fact that Fiala, Kamiński and Paulusma settled the computational complexity of H-Induced Minor for all forests on at most 7 vertices, except for one case. An other interesting open problem regarding the complexity of H-Induced Minor for some small fixed graph H is the following:

Question 3. Can $2C_3$-Induced Minor be solved in polynomial time?

The graph $2C_3$ is isomorphic to two disjoint cycles of length 3. The problem is equivalent to determining whether some input graph G has two mutually induced cycles, i.e., two vertex-disjoint cycles that do not contain adjacent vertices. In addition, this appears to be the smallest graph H for which the complexity of H-Induced Minor is unknown.

As we saw in Section 3.7, Fellows, Kratochvíl, Middendorf and Pfeiffer proved that H-Induced Minor can be solved in polynomial time on planar graphs. They subsequently ask the following natural question:

Question 4. Can H-Induced Minor be solved in polynomial time on graphs of bounded genus?

Following the introduction of the lift-contraction and lift-minor operations in [71], it is natural to ask about the computational complexity of deciding these two relations:

Question 5. What is the complexity of H-Lift-contraction and H-Lift-Minor? In particular, are there some fixed graphs H for which H-Lift-contraction or H-Lift-Minor is NP-complete?

As we mentioned in Section 3.4, both Lift-contraction and Lift-Minor are NP-complete, therefore only the case where H is fixed needs to be considered.

The following questions follow from the papers constituting the scientific contribution (Part II) of this thesis:

Question 6. Can Contractibility be solved in polynomial time on proper interval graphs?

Note that we mention only Contractibility in Question 9 since, as we saw in Propositions 3.15 and 3.16, Subgraph, Minor, Topological Minor, Immersion and their induced counterparts are all NP-complete when G and H are restricted to be disjoint unions of paths.

Question 6 follows naturally from the results of [10], presented in Chapter 6, stating that Contractibility can be solved in polynomial-time when G is a trivially perfect graph and H is a threshold graph part of the input, while they become NP-complete when both G and H are trivially perfect graphs. Since trivially perfect graphs are a subclass of interval graphs, this implies that Contractibility is NP-complete on interval graphs. Moreover, recall that it was shown in [7], which is presented in Chapter 7, that H-Contractibility, H-Induced Minor and H-Induced Topological Minor can be solved in polynomial time on chordal graphs, and hence on interval graphs. Whether this result is tight is certainly a natural question to ask:

Question 7. Are H-Contractibility, H-Induced Minor and H-Induced Topological Minor FPT on interval graphs?

Moreover, the polynomial-time solvability of H-Contractibility and H-Induced Minor on chordal graphs was achieved by using a polynomial time algorithm for the k-Set Restricted Disjoint Paths problem as a sub-routine. Therefore it is natural to ask whether k-Set Restricted Disjoint Paths can be solved faster on chordal graphs:

Question 8. Is k-Set Restricted Disjoint Paths FPT on chordal graphs?

Note that it was mentioned by Golovach, Kratsch and Paulusma that the problem is FPT on interval graphs [121]. Moreover, it is interesting to note that, while the algorithm for H-Contractibility when G is a split graph described in [10] runs in time $|V(G)|^{O(|V(H)|)}$, the algorithm for the same problem when G is chordal given in [7] runs in time $|V(G)|^{O(|V(H)|^2)}$. We ask the following question:

Question 9. Can H-Contractibility and H-Induced Minor be solved in time in time $|V(G)|^{O(|V(H)|)}$ on chordal graphs?

In [12], presented in Chapter 8, we showed that the problem H-Induced Immersion can be solved in polynomial-time for every fixed target multigraph H. The natural follow-up question is therefore:

Question 10. Is H-Induced Immersion FPT?

As mentioned in [12], there is a polynomial-time reduction from H-Immersion to H-Induced Immersion, which means that the answer to Question 10 would imply the FPT algorithm for H-Immersion given in [79], which strongly suggests that it would be difficult to provide a positive answer to Question 10. Similarly, Robertson and Seymour proved Nash-William's immersion conjecture in [132], which states that the set of all graphs is well-quasi-ordered under weak immersion. Asking if the same applies to induced immersion follows naturally[2]:

Question 11. Are graphs well-quasi-ordered under induced immersion?

Note that Fellows, Hermelin and Rosamond asked a similar question in [54].

Again, providing a positive answer to this question would imply Nash-William's immersion conjecture. In the same idea of extending results about immersion, we have the following question:

[2]It is possible to define a "strong" variant of induced immersion by forcing, for every lift $\{uv, vw\}$, to delete the vertex v at the end of the sequence of operations.

Question 12. Is there a constant c_h for every planar subcubic multigraph H such that H-induced immersion free graphs have treewidth at most c_H?

Together with an MSO-expression for induced immersion, a positive answer to Question 11 would imply that H-Induced Immersion can be solved in linear-time whenever H is a planar subcubic multigraph. We believe that the answer to both Question 11 and whether induced immersion is MSO-expressible is positive. The last problem we would like to mention is about the possible extension of the main result of [14], presented in Chapter 10, namely the characterization of graph of carving-width at 3:

Question 13. What is the structure of graphs having carving-width at most 4? Alternatively, what is the set of topological minor obstructions (or immersion obstructions) for graphs of carving-width at most 4?

If such a characterization is obtained, it would be most interesting to use it to derive a practical and efficient linear-time algorithm to recognize graphs of carving-width at most 4.

5.3 Overview of Part II

Finally, we would like to conclude Part I of this thesis by giving a short overview of Part II. As mentioned earlier, the remainder of this thesis is dedicated to the results that constitute the scientific contribution of this thesis. The second part of this thesis contains 5 chapters, each consisting of one of the following papers:

5.3.1 Chapter 6

Edge contractions in subclasses of chordal graphs [11].
Rémy Belmonte, Pinar Heggernes and Pim van 't Hof.
Discrete Applied Mathematics 160(7-8), pp. 999-1010 (2012).
A preliminary version appeared at TAMC 2011 [10].

We study the complexity of the Contractibility problem on the classes of split, threshold and trivially perfect graphs. We show that the problem is NP-complete in two cases: when G is split and H is threshold, and

when both G and H are trivially perfect. It is also shown that Induced Subgraph Isomorphism is NP-complete when both G and H are trivially perfect. Moreover, it is shown that Contractibility can be solved in cubic time when G is a trivially perfect graph and H is threshold, and in linear time when G is restricted to be a threshold graph. We would like to point out that Theorem 1 in Chapter 6 implies that for any two trivially perfect graphs G, H, the following are equivalent:

(i) G contains H as an induced subgraph;

(ii) G contains H as induced minor;

(iii) G contains H as an induced topological minor.

Therefore, all results regarding NP-completeness and polynomial time solvability of the problems Induced Subgraph Isomorphism and Contractibility on trivially perfect graphs immediately translate into equivalent results for Induced Minor and Induced Topological Minor. Moreover, the NP-completeness proof for the case where G is split and H is threshold (Theorem 7) can easily be modified to obtain the same result for Induced Minor. Finally, we study the complexity of H-Contractibility when G is a split graph and prove that the problem can be solved in time $f(|V(H)|) \cdot |V(G)|^{O(\alpha(H))}$, where $\alpha(H)$ is the size of maximum independent set in H. Note that this means the problem H-Contractibility is FPT when parameterized by $|V(H)|$ if the size of a maximum independent set of H is bounded from above by a constant.

5.3.2 Chapter 7

Detecting fixed patterns in chordal graphs in polynomial time [7].
Rémy Belmonte, Petr A. Golovach, Pinar Heggernes, Pim van 't Hof, Marcin Kamiński and Daniël Paulusma.
Algorithmica, to appear (2013).
A preliminary version appeared at ISAAC 2011 [8].

We study the complexity of various containment relations on chordal graphs, thereby extending the polynomial time solvability result of H-Contractibility from Chapter 6 and several other results from [71]. We give a complete classification in terms of classical and parameterized complexity of all the containment relations obtained by combining the following operations: vertex deletion, edge deletion, edge contraction and

vertex dissolution. Our main result is to show that H-Induced Minor, H-Contractibility and H-Induced Topological Minor can be solved in polynomial time on chordal graphs. In order to obtain these results, we introduce set restricted variants of the k-Disjoint Paths and k-Induced Disjoint Paths problems. We prove that the former is NP-complete on general graphs already when $k = 2$, but solvable in polynomial time for every fixed k on chordal graphs, and that the latter is polynomial time solvable on chordal graphs, even when k is part of the input. As a byproduct, we obtain a polynomial time algorithm for the k-Set Disjoint Connected Subgraphs problem on interval graphs for every fixed k.

5.3.3 Chapter 8

Induced immersions [12].
Rémy Belmonte, Pim van 't Hof and Marcin Kamiński.
In *Proceedings of ISAAC 2012*, LNCS 7676: 299–308, Springer (2012).

We initiate the study of the induced immersion relation. We prove that the Induced Immersion problem is NP-complete on planar graphs of maximum degree at most 4, while H-Induced Immersion can be solved in polynomial time for every input simple graph G and fixed multigraph H. We also give a polynomial time reduction from the Immersion problem to the Induced Immersion problem, which implies that providing an FPT algorithm for H-Induced Immersion would extend the algorithm for H-Immersion from [79]. Finally, we study the structure of graphs that do not contain a fixed graph H as an induced immersion. We first show that for every multigraph H, there exists a graph H' such that the class of H-induced immersion free graphs is contained in the class of H'-immersion free graphs. This particularly implies that all structural properties of H'-immersion free graphs, such as the decomposition theorem of Wollan [152] can be applied to H-immersion free graphs. We make use of this result by showing that multigraphs that exclude a fixed multigraph of maximum degree at most 2 as an induced immersion have bounded treewidth.

5.3.4 Chapter 9

Structure of W_4-immersion free graphs [6].
Rémy Belmonte, Archontia Giannopoulou, Daniel Lokshtanov and Dimitrios M. Thilikos.
Submitted (2013).

We study the structure of graphs that do not contain the graph W_4, i.e., the wheel on five vertices, as an immersion. We prove that these graphs can be decomposed via so-called internal edge-sums of order at most 3 in such a way that the prime components of the decomposition are either subcubic or have bounded treewidth.

5.3.5 Chapter 10

Characterizing graphs of small carving-width [14].
Rémy Belmonte, Pim van 't Hof, Marcin Kamiński, Daniël Paulusma and Dimitrios M. Thilikos.
Discrete Applied Mathematics 161(13-14), pp. 1888-1893 (2013).
A preliminary version appeared at COCOA 2012 [13].

We study the structure of graphs having carving-width at most 3. The width parameter carving-width was introduced by Seymour and Thomas in [140], where they proved that deciding if a graph has carving-width at most k is NP-complete when k is part of the input. Later, it was shown by Thilikos, Serna and Bodlaender [143] that it can be decided in linear-time whether a graph has carving width at most k for every fixed k. However, the constants involved in their algorithm are enormous, and it is therefore necessary to design more efficient and practical algorithms for small values of k. To that end, we study the structure of graphs having carving-width at most 3. We prove that:

- A graph has carving-width at most 1 if and only if it has maximum degree at most 1.
- A graph has carving-width at most 2 if and only if it has maximum degree at most 2.
- A graph has carving-width at most 3 if and only if it has maximum degree at most 3 and treewidth at most 2.

As a byproduct, we obtain the topological minor obstructions for graphs of carving-width at most 3. We also give some counter examples to a possible natural extension of the result to graphs of carving-width at most 4 of the form "A graph as carving-width at most 4 if and only if it has maximum degree at most 4 and treewidth at most 3".

Bibliography

[1] Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *SODA*, pages 641–650, 2008.

[2] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *Journal of the ACM*, 51(3):363–384, 2004.

[3] Stefan Arnborg, Andrzej Proskurowski, and Derek G. Corneil. Forbidden minors characterization of partial 3-trees. *Discrete Mathematics*, 80(1):1–19, 1990.

[4] Gábor Bacsó and Zsolt Tuza. A characterization of graphs without long induced paths. *Journal of Graph Theory*, 14(4):455–464, 1990.

[5] Gabor Bacsó and Zsolt Tuza. Dominating cliques in P_5-free graphs. *Periodica Mathematica Hungarica*, 21(4):303–308, 1990.

[6] Rémy Belmonte, Archontia C. Giannopoulou, Daniel Lokshtanov, and Dimitrios M. Thilikos. Structure of W_4-immersion free graphs. Submitted.

[7] Rémy Belmonte, Petr A. Golovach, Pinar Heggernes, Pim van 't Hof, Marcin Kamiński, and Daniël Paulusma and. Detecting fixed patterns in chordal graphs in polynomial time. To appear in Algorithmica.

[8] Rémy Belmonte, Petr A. Golovach, Pinar Heggernes, Pim van 't Hof, Marcin Kaminski, and Daniël Paulusma. Finding contractions and induced minors in chordal graphs via disjoint paths. In *ISAAC*, pages 110–119, 2011.

[9] Rémy Belmonte, Petr A. Golovach, Pim van 't Hof, and Daniël Paulusma. Parameterized complexity of two edge contraction problems with degree constraints. To appear at IPEC 2013.

[10] Rémy Belmonte, Pinar Heggernes, and Pim van 't Hof. Edge contractions in subclasses of chordal graphs. In *TAMC*, pages 528–539, 2011.

[11] Rémy Belmonte, Pinar Heggernes, and Pim van 't Hof. Edge contractions in subclasses of chordal graphs. *Discrete Applied Mathematics*, 160(7-8):999–1010, 2012.

[12] Rémy Belmonte, Pim van 't Hof, and Marcin Kamiński. Induced immersions. In *ISAAC*, pages 299–308, 2012.

[13] Rémy Belmonte, Pim van 't Hof, Marcin Kaminski, Daniël Paulusma, and Dimitrios M. Thilikos. Characterizing graphs of small carving-width. In *COCOA*, pages 360–370, 2012.

[14] Rémy Belmonte, Pim van 't Hof, Marcin Kamiński, Daniël Paulusma, and Dimitrios M. Thilikos. Characterizing graphs of small carving-width. *Discrete Applied Mathematics*, 161(13-14):1888–1893, 2013.

[15] Daniel Bienstock, Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a forest. *Journal of Combinatorial Theory, Series B*, 52(2):274–283, 1991.

[16] Etienne Birmelé, John A. Bondy, and Bruce A. Reed. Brambles, prisms and grids. In *Graph Theory in Paris*, Trends in Mathematics, pages 37–44. Birkhäuser Basel, 2007.

[17] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[18] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 09(1-2):1–45, 1998.

[19] André Bouchet. Transforming trees by successive local complementations. *Journal of Graph Theory*, 12(2):195–207, 1988.

[20] André Bouchet. An efficient algorithm to recognize locally equivalent graphs. *Combinatorica*, 11(4):315–329, 1991.

[21] André Bouchet. Circle graph obstructions. *Journal of Combinatorial Theory, Series B*, 60(1):107–144, 1994.

[22] Andreas Brandstädt, Van Bang Le, and Jeremy Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999.

[23] Andries E. Brouwer and Henk J. Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11(1):71–79, 1987.

[24] Kevin Cattell and Michael J. Dinneen. A characterization of graphs with vertex cover up to five. In *ORDAL*, pages 86–99, 1994.

[25] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. arXiv e-print 1305.6577, 2013.

[26] Yijia Chen and Jörg Flum. On parameterized path and chordless path problems. In *IEEE Conference on Computational Complexity*, pages 250–263, 2007.

[27] Maria Chudnovsky. The structure of bull-free graphs II and III - a summary. *Journal of Combinatorial Theory, Series B*, 102(1):252–282, 2012.

[28] Maria Chudnovsky, Alexandra Ovetsky Fradkin, and Paul D. Seymour. Tournament immersion and cutwidth. *Journal of Combinatorial Theory, Series B*, 102(1):93–101, 2012.

[29] Maria Chudnovsky, Irena Penev, Alex Scott, and Nicolas Trotignon. Excluding induced subdivisions of the bull and related graphs. *Journal of Graph Theory*, 71(1):49–68, 2012.

[30] Maria Chudnovsky and Paul D. Seymour. Claw-free graphs. IV. decomposition theorem. *Journal of Combinatorial Theory, Series B*, 98(5):839–938, 2008.

[31] Václáv Chvátal and Peter L. Hammer. Set-packing and threshold graphs. University of Waterloo Research Reports, CORR 73-21, 1973.

[32] Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.

[33] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Compututation*, 85(1):12–75, 1990.

[34] Bruno Courcelle and Jost Engelfriet. *Graph structure and monadic second-order logic, a language theoretic approach*. Cambridge University Press, 2012.

[35] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[36] Peter Damaschke. Induced subgraph isomorphism for cographs in NP-complete. In *WG*, pages 72–78, 1990.

[37] Peter Damaschke. Induced subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 14(4):427–435, 1990.

[38] H.N. de Ridder et al. Information system on graph classes and their inclusions (ISGCI). http://www.graphclasses.org, 2001-2013.

[39] Matt DeVos, Jessica McDonald, Bojan Mohar, and Diego Scheide. A note on forbidding clique immersions. arXiv e-print 1207.2117, 2012.

[40] Reinhard Diestel. *Graph Theory*. Springer, 2010.

[41] Reinhard Diestel, Tommy R. Jensen, Konstantin Yu. Gorbunov, and Carsten Thomassen. Highly connected sets and the excluded grid theorem. *Journal of Combinatorial Theory, Series B*, 75(1):61–73, 1999.

[42] Guoli Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16(5):489–502, 1992.

[43] Guoli Ding. Chordal graphs, interval graphs, and wqo. *Journal of Graph Theory*, 28(2):105–114, 1998.

[44] Guoli Ding. On canonical antichains. *Discrete Mathematics*, 309(5):1123–1134, 2009.

[45] Michael J. Dinneen, Kevin Cattell, and Michael R. Fellows. Forbidden minors to graphs with small feedback sets. *Discrete Mathematics*, 230(1-3):215–252, 2001.

[46] Michael J. Dinneen and Ralph Versteegen. Obstructions for the graphs of vertex cover seven. *CDMTCS Research Report Series*, 2012.

[47] Michael J. Dinneen and Liu Xiong. Minor-order obstructions for the graphs of vertex cover 6. *Journal of Graph Theory*, 41(3):163–178, 2002.

[48] Rod G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[49] Zdenek Dvorak. A stronger structure theorem for excluded topological minors. arXiv e-print 1209.0129, 2012.

[50] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3), 1999.

[51] David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.

[52] David Eppstein. Finding large clique minors is hard. *Journal of Graph Algorithms and Applications*, 13(2):197–204, 2009.

[53] Graham Farr. The subgraph homeomorphism problem for small wheels. *Discrete Mathematics*, 71(2):129–142, 1988.

[54] Michael R. Fellows, Danny Hermelin, and Frances A. Rosamond. Well quasi orders in subclasses of bounded treewidth graphs and their algorithmic applications. *Algorithmica*, 64(1):3–18, 2012.

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

[56] Michael R. Fellows and Michael A. Langston. On search, decision, and the efficiency of polynomial-time algorithms. *Journal of Computer and System Sciences*, 49(3):769–779, 1994.

[57] Jirí Fiala, Marcin Kaminski, and Daniël Paulusma. Detecting induced star-like minors in polynomial time. *Journal of Discrete Algorithms*, 17:74–85, 2012.

[58] Jirí Fiala, Marcin Kamiński, and Daniël Paulusma. A note on contracting claw-free graphs. *Discrete Mathematics & Theoretical Computer Science*, 2013. To appear.

[59] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[60] Stephane Földes and Peter L. Hammer. Split graphs. In *Proceedings of the 8th southeastern Conference on Combinatorics, graph theory, and computing*, pages 311–315, 1977.

[61] Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[62] Alexandra Ovetsky Fradkin and Paul D. Seymour. Tournament pathwidth and topological containment. *Journal of Combinatorial Theory, Series B*, 103(3):374–384, 2013.

[63] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences, 1979.

[64] Michael R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[65] James F. Geelen, A. M. H. Gerards, and Geoff Whittle. Branch-width and well-quasi-ordering in matroids and graphs. *Journal of Combinatorial Theory, Series B*, 84(2):270–290, 2002.

[66] Jim Geelen, Bert Gerards, and Geoff Whittle. Excluding a planar graph from gf(q)-representable matroids. *Journal of Combinatorial Theory, Series B*, 97(6):971–998, 2007.

[67] Jim Geelen and Sang il Oum. Circle graph obstructions under pivoting. *Journal of Graph Theory*, 61(1):1–11, 2009.

[68] Archontia C. Giannopoulou, Iosif Salem, and Dimitris Zoros. Effective computation of immersion obstructions for unions of graph classes. In *SWAT*, pages 165–176, 2012.

[69] H. H. Glover and J. P. Huneke. There are Ã£nitely many kuratowski graphs for the projective plane. In *Graph Theory and Related Topics*, pages 201–206, 1979.

[70] Petr A. Golovach, Pinar Heggernes, Pim van 't Hof, Fredrik Manne, Daniël Paulusma, and Michal Pilipczuk. How to eliminate a graph. In *WG*, pages 320–331, 2012.

[71] Petr A. Golovach, Marcin Kaminski, Daniël Paulusma, and Dimitrios M. Thilikos. Lift contractions. *Electronic Notes in Discrete Mathematics*, 38:407–412, 2011.

[72] Petr A. Golovach, Marcin Kaminski, Daniël Paulusma, and Dimitrios M. Thilikos. Containment relations in split graphs. *Discrete Applied Mathematics*, 160(1-2):155–163, 2012.

[73] Petr A. Golovach, Dieter Kratsch, and Daniël Paulusma. Detecting induced minors in AT-free graphs. *Theoretical Computer Science*, 482:20–32, 2013.

[74] Petr A. Golovach, Bernard Lidický, Barnaby Martin, and Daniël Paulusma. Finding vertex-surjective graph homomorphisms. In *CSR*, pages 160–171, 2012.

[75] Petr A. Golovach, Daniël Paulusma, and Erik Jan van Leeuwen. Induced disjoint paths in at-free graphs. In *SWAT*, pages 153–164, 2012.

[76] Petr A. Golovach, Daniël Paulusma, and Erik Jan van Leeuwen. Induced disjoint paths in claw-free graphs. In *ESA*, pages 515–526, 2012.

[77] Martin Charles Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24:105–107, 1978.

[78] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier, 2006.

[79] Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *STOC*, pages 479–488, 2011.

[80] Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In *STOC*, pages 173–192, 2012.

[81] Martin Handford. *The Great Waldo Search*. Little Brown & Co, 1989.

[82] Pavol Hell and Jaroslav Nesetril. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92–110, 1990.

[83] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, Oxford, 2004.

[84] Danny Hermelin, Matthias Mnich, Erik Jan van Leeuwen, and Gerhard J. Woeginger. Domination when the stars are out. In *ICALP (1)*, pages 462–473, 2011.

[85] Petr Hlinený, Sang il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *The Computer Journal*, 51(3):326–362, 2008.

[86] Sang il Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B*, 95(1):79–100, 2005.

[87] Marcin Kaminski, Daniël Paulusma, and Dimitrios M. Thilikos. Contractions of planar graphs in polynomial time. In *ESA (1)*, pages 122–133, 2010.

[88] Marcin Kaminski, Daniël Paulusma, and Dimitrios M. Thilikos. Contracting planar graphs to contractions of triangulations. *Journal of Discrete Algorithms*, 9(3):299–306, 2011.

[89] Marcin Kaminski and Dimitrios M. Thilikos. Contraction checking in graphs on surfaces. In *STACS*, pages 182–193, 2012.

[90] Richard M. Karp. On the complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[91] Ken-ichi Kawarabayashi and Yusuke Kobayashi. The induced disjoint paths problem. In *IPCO*, pages 47–61, 2008.

[92] Ken-ichi Kawarabayashi and Yusuke Kobayashi. Linear min-max relation between the treewidth of h-minor-free graphs and its largest grid. In *STACS*, pages 278–289, 2012.

[93] Ken-ichi Kawarabayashi and Yusuke Kobayashi. A linear time algorithm for the induced disjoint paths problem in planar graphs. *Journal of Computer and System Sciences*, 78(2):670–680, 2012.

[94] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

[95] Ken-ichi Kawarabayashi and Paul Wollan. A simpler algorithm and shorter proof for the graph minor decomposition. In *STOC*, pages 451–458, 2011.

[96] Shuji Kijima, Yota Otachi, Toshiki Saitoh, and Takeaki Uno. Subgraph isomorphism in graph classes. *Discrete Mathematics*, 312(21):3164–3173, 2012.

[97] Nancy G. Kinnersley. *Obstruction set isolation for layout permutation problems*. PhD thesis, Department of Computer Science, Washington State University, 1989.

[98] Nancy G. Kinnersley and Michael A. Langston. Obstruction set isolation for the gate matrix layout problem. *Discrete Applied Mathematics*, 54(2-3):169–213, 1994.

[99] Nicholas Korpelainen and Vadim V. Lozin. Bipartite induced subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 67(3):235–249, 2011.

[100] J. B. Kruskal. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95(2):pp. 210–225, 1960.

[101] Kazimierz Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[102] Lap Chi Lau and Chun Kong Yung. Efficient edge splitting-off algorithms maintaining all-pairs edge-connectivities. In *IPCO*, pages 96–109, 2010.

[103] Alexander Leaf and Paul Seymour. The structure of graphs not admitting a fixed immersion. Submitted for publication.

[104] C. G. Lekkerkerker and J. Ch. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematica*, 51:45–64, 1962.

[105] Benjamin Lévêque, David Y. Lin, Frédéric Maffray, and Nicolas Trotignon. Detecting induced subgraphs. *Discrete Applied Mathematics*, 157(17):3540–3551, 2009.

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

[107] Asaf Levin, Daniël Paulusma, and Gerhard J. Woeginger. The computational complexity of graph contractions II: Two tough polynomially solvable cases. *Networks*, 52(1):32–56, 2008.

[108] Jiping Liu, Yuejian Peng, and Cheng Zhao. Characterization of P_6-free graphs. *Discrete Applied Mathematics*, 155(8):1038–1043, 2007.

[109] László Lovász. Lecture. Conference of Graph Theory, Prague, 1974.

[110] László Lovász. *Combinatorial problems and exercises*. Elsevier, 1979.

[111] Vadim V. Lozin and Colin Mayhill. Canonical antichains of unit interval and bipartite permutation graphs. *Order*, 28(3):513–522, 2011.

[112] Dániel Marx and Ildikó Schlotter. Cleaning interval graphs. *Algorithmica*, 65(2):275–316, 2013.

[113] Jirí Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.

[114] C. St. J. A. Nash-Williams. On well-quasi-ordering trees. In *Theory of Graphs and Its Applications (Proc. Symp. Smolenice, 1963)*, pages 83–84, 1964.

[115] Eugene T. Neufeld. Practical toroidality testing. Master's thesis, University of Victoria, 1993.

[116] Jaroslav Nešetřil and Patrice Ossona de Mendez. Grad and classes with bounded expansion II. algorithmic aspects. *European Journal of Combinatorics*, 29(3):777–791, 2008.

[117] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[118] Stavros D. Nikolopoulos and Leonidas Palios. Detecting holes and antiholes in graphs. *Algorithmica*, 47(2):119–138, 2007.

[119] James Oxley. *Matroid Theory*. Oxford Graduate Texts in Mathematics. Oxford University Press, second edition, 2011.

[120] S. Parter. The use of linear graphs in gauss elimination. *SIAM Review*, 3(2):pp. 119–130, 1961.

[121] Daniël Paulusma Peter A. Golovach, Dieter Kratsch. Private communication, 2012.

[122] Michal Pilipczuk. Computing cutwidth and pathwidth of semicomplete digraphs via degree orderings. In *STACS*, pages 197–208, 2013.

[123] Jean-Florent Raymond and Dimitrios M. Thilikos. Low polynomial exclusion of planar graph patterns. arXiv e-print 1305.7112, 2013.

[124] F. S. Roberts. Indifference graphs. In *Proof Techniques in Graph Theory, Proceedings of the 2nd Annual Arbor Graph Theory Conference*, pages 139–146, 1969.

[125] Neil Robertson and Paul D. Seymour. Graph minors. I. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.

[126] Neil Robertson and Paul D. Seymour. Graph minors. V. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.

[127] Neil Robertson and Paul D. Seymour. Graph minors. IV. tree-width and well-quasi-ordering. *Journal of Combinatorial Theory, Series B*, 48(2):227–254, 1990.

[128] Neil Robertson and Paul D. Seymour. Graph minors. VIII. a kuratowski theorem for general surfaces. *Journal of Combinatorial Theory, Series B*, 48(2):255–288, 1990.

[129] Neil Robertson and Paul D. Seymour. Graph minors .XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[130] Neil Robertson and Paul D. Seymour. Graph minors. XVI. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76, 2003.

[131] Neil Robertson and Paul D. Seymour. Graph minors. XX. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[132] Neil Robertson and Paul D. Seymour. Graph minors XXIII. nash-williams' immersion conjecture. *Journal of Combinatorial Theory, Series B*, 100(2):181–205, 2010.

[133] Neil Robertson, Paul D. Seymour, and Robin Thomas. Hadwiger's conjecture for k $_6$-free graphs. *Combinatorica*, 13(3):279–361, 1993.

[134] Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994.

[135] Rebecca J. Robinson. *Characterizations and algorithms for topological containment of wheel graphs*. PhD thesis, Monash University. Faculty of Information Technology. Clayton School of Information Technology, 2009.

[136] Rebecca J. Robinson and Graham Farr. Structure and recognition of graphs with no 6-wheel subdivision. *Algorithmica*, 55(4):703–728, 2009.

[137] Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, 1988.

[138] Alex D. Scott. Induced trees in graphs of large chromatic number. *Journal of Graph Theory*, 24(4):297–311, 1997.

[139] Paul D. Seymour. Disjoint paths in graphs. *Discrete Mathematics*, 29(3):293–309, 1980.

[140] Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[141] Yossi Shiloach. A polynomial solution to the undirected two paths problem. *Journal of the ACM*, 27(3):445–456, 1980.

[142] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[143] Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Constructive linear time algorithms for small cutwidth and carving-width. In *ISAAC*, pages 192–203, 2000.

[144] Robin Thomas. Graphs without k_4 and well-quasi-ordering. *Journal of Combinatorial Theory, Series B*, 38(3):240–247, 1985.

[145] Carsten Thomassen. 2-linked graphs. *European Journal of Combinatorics*, 1:371–378, 1980.

[146] Pim van 't Hof, Marcin Kaminski, Daniël Paulusma, Stefan Szeider, and Dimitrios M. Thilikos. On graph contractions and induced minors. *Discrete Applied Mathematics*, 160(6):799–809, 2012.

[147] Pim van 't Hof and Daniël Paulusma. A new characterization of p_6-free graphs. *Discrete Applied Mathematics*, 158(7):731–740, 2010.

[148] Klaus Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

[149] G. Wegner. *Eigenschaften der Nerven homologisch–einfacher Familien im R^n*. PhD thesis, University of Göttingen, 1967.

[150] E. S. Wolk. The comparability graph of a tree. *Proceedings of the American Mathematical Society*, 13:789–795, 1962.

[151] E. S Wolk. A note on "the comparability graph of a tree". *Proceedings of the American Mathematical Society*, 16:17–20, 1965.

[152] Paul Wollan. The structure of graphs not admitting a fixed immersion. arXiv e-print 1302.3867, 2013.

[153] Chun Kong Yung. Edge splitting-off and network design problems. Master's thesis, Department of Computer Science and Engineering, The Chinese University of Hong Kong, 2009.

Part II

New results

Chapter 6

Edge Contractions in Subclasses of Chordal Graphs

Edge Contractions in Subclasses of Chordal Graphs

Rémy Belmonte, Pinar Heggernes, Pim van 't Hof

Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
{remy.belmonte,pinar.heggernes,pim.vanthof}@ii.uib.no

Abstract

Modifying a given graph to obtain another graph is a well-studied problem with applications in many fields. Given two input graphs G and H, the Contractibility problem is to decide whether H can be obtained from G by a sequence of edge contractions. This problem is known to be NP-complete already when both input graphs are trees of bounded diameter. We prove that Contractibility can be solved in polynomial time when G is a trivially perfect graph and H is a threshold graph, thereby giving the first classes of graphs of unbounded treewidth and unbounded degree on which the problem can be solved in polynomial time. We show that this polynomial-time result is in a sense tight, by proving that Contractibility is NP-complete when G and H are both trivially perfect graphs, and when G is a split graph and H is a threshold graph. If the graph H is fixed and only G is given as input, then the problem is called H-Contractibility. This problem is known to be NP-complete on general graphs already when H is a path on four vertices. We show that, for any fixed graph H, the H-Contractibility problem can be solved in polynomial time if the input graph G is a split graph.

1 Introduction

The problem of deciding whether a given graph can be obtained from another given graph by contracting edges is motivated by Hamiltonian graph theory and graph minor theory, and it has applications in computer graphics and cluster analysis [19]. This problem has recently attracted increasing interest, in particular when restrictions are imposed on the

input graphs [5, 17, 18, 19, 20]. We continue this line of research with new polynomial-time and NP-completeness results.

For a fixed graph H, the H-Contractibility problem is to decide whether H can be obtained from an input graph G by a sequence of edge contractions. This problem is closely related to the well-known H-Minor Containment problem, which is the problem of deciding whether H can be obtained from a *subgraph* of G by contracting edges. A celebrated result by Robertson and Seymour [24] states that H-Minor Containment can be solved in polynomial time on general graphs for any fixed H. As a contrast, H-Contractibility is NP-complete already for very simple fixed graphs H, such as a path or a cycle on four vertices [5]. The version of the problem where both graphs are given as input, called Contractibility, is NP-complete on trees of bounded diameter, as well as on trees in which at most one vertex has degree more than 3 [23].

In this paper, we study the Contractibility and H-Contractibility problems on subclasses of chordal graphs. All the graph classes that are mentioned in this paper, as well as the inclusion relationships between the different classes, are depicted in Figure 1. Chordal graphs constitute one of the most famous graph classes, with a large number of practical applications (see e.g., [11, 14, 25]). It is easy to see, for example using the well-known characterization of chordal graphs as the intersection graphs of subtrees in a tree [10], that edge contractions preserve the property of being chordal; contracting an edge in a chordal graph is equivalent to "merging" two subtrees in the intersection model. Since trees are chordal graphs, it follows from the above-mentioned hardness result on trees that Contractibility is NP-complete when G and H are both chordal. We show that the problem remains NP-complete even when G and H are both trivially perfect graphs or both split graphs. Note that trees do not form a subclass of trivially perfect graphs and also not of split graphs. Trivially perfect graphs and split graphs are two unrelated subclasses of chordal graphs, and both classes are well-studied with several theoretical applications [4, 14]. These two classes share a common subclass called threshold graphs, which is another well-known subclass of chordal graphs [22]. We prove that Contractibility remains NP-complete even when G is split and H is threshold.

On the positive side, we show that Contractibility can be solved in polynomial time when G is trivially perfect and H is threshold. This result can be considered tight by the above-mentioned hardness results. For H-Contractibility, we give a polynomial-time algorithm when G is a split graph and H is an arbitrary fixed graph. Our algorithm runs in time $f(|V(H)|) \cdot |V(G)|^{O(\alpha(H))}$, where $\alpha(H)$ denotes the size of

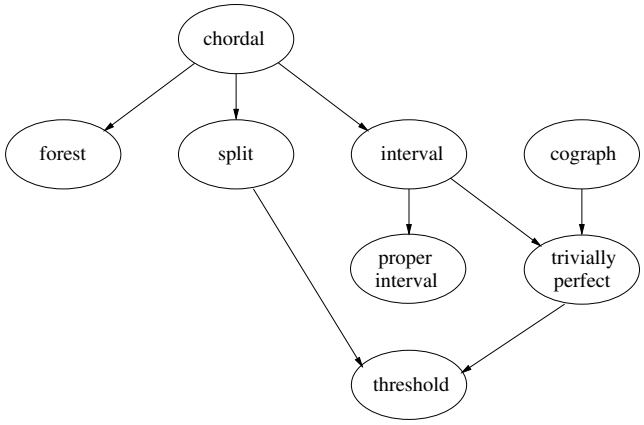


Figure 1: The graph classes mentioned in this paper, where $\rightarrow$ represents the $\supset$ relation.

G	H	Complexity
Trivially perfect (i)	Trivially perfect (i)	NP-complete
Trivially perfect (i)	Threshold (i)	Polynomial
Trivially perfect (i)	Trivially perfect (f)	Polynomial
Threshold (i)	Arbitrary (i)	Linear
Split (i)	Threshold (i)	NP-complete
Split (i)	Arbitrary (f)	Polynomial

Table 1: The complexity of deciding whether G can be contracted to H, according to our results; (i) stands for "part of the input", (f) stands for "fixed".

a maximum independent set in H, and f is some function that does not depend on the size of G. Very recently, Contractibility was shown to be $W[1]$-hard on split graphs when parameterized by $|V(H)|$ [13], which implies that it is highly unlikely that this problem can be solved in time $f(|V(H)|) \cdot |V(G)|^{O(1)}$ on split graphs (see [8] for the definition of $W[1]$-hardness and more details on parameterized complexity). This makes our polynomial-time algorithm for H-Contractibility on split graphs in some sense tight. Our results on Contractibility and H-Contractibility presented in this paper are summarized in Table 1.

As an interesting byproduct of our results, we show that the problems Contractibility and Induced Subgraph Isomorphism are equivalent on connected trivially perfect graphs. Hence our results imply that the latter problem is NP-complete on connected trivially perfect graphs,

and that this problem can be solved in polynomial time when G is trivially perfect and H is threshold. We would like to mention that Induced Subgraph Isomorphism is known to be NP-complete on split graphs and on cographs [6]. Trivially perfect graphs constitute a subclass of cographs, and threshold graphs are both cographs and split graphs. Hence our results tighten previously known hardness results on Induced Subgraph Isomorphism.

To finish this section, let us mention some related work. Both Contractibility and H-Contractibility have been studied on special graph classes before. Given the previously mentioned NP-completeness results of Contractibility on some subclasses of trees, it is perhaps not surprising that hardly any positive results are known for this problem. Prior to our work, Contractibility was known to be solvable in polynomial time only when G has bounded treewidth and H has bounded degree [23]. A few more positive results are known on the H-Contractibility problem. For example, for every fixed graph H on at most 5 vertices, H-Contractibility can be solved on general graphs in polynomial time when H has a universal vertex, and it is NP-complete otherwise [19, 20]. However, it is known that for larger fixed graphs H, the presence of a universal vertex in H is not a guarantee for polynomial-time solvability of the problem [17]. On planar input graphs, H-Contractibility can be solved in polynomial time for every fixed graph H [18]. As very recent work, after our results were first announced at TAMC 2011 [3], Golovach, Kamiński and Paulusma [12] showed that H-Contractibility can be solved in polynomial time on chordal graphs for any fixed split graph H, as well as for any fixed tree H. This was then extended by Belmonte et al. [2], who showed that H-Contractibility can be solved in polynomial time on chordal graphs for *any* fixed graph H. The mentioned results of [12] and [2] imply algorithms for H-Contractibility on split graphs that run in time $|V(G)|^{O(|V(H)|^2)}$. An algorithm for H-Contractibility on split graphs with running time $|V(G)|^{O(|V(H)|)}$ has also been announced simultaneously by Golovach et al. [13]. As we will see in Section 4, the asymptotically better running time of our algorithm is obtained by using structural properties of split graphs that are contractible to a fixed graph H.

2 Preliminaries

All graphs considered in this paper are undirected, finite and simple. For terminology not defined below, we refer the reader to any general graph

theory textbook, for example the one by Diestel [7]. More information on the graph classes mentioned in this paper, including a wealth of information on applications of these classes, can be found in the monograph by Golumbic [14].

For a graph G, we use $V(G)$ and $E(G)$ to denote the set of vertices and set of edges of G, respectively. Let G be a graph, and let $V = V(G)$ and $E = E(G)$. For a vertex v in G, the set $N_G(v) = \{w \in V \mid vw \in E\}$, consisting of all the neighbors of v in G, is called the *neighborhood* of v. The set $N_G[v] = N_G(v) \cup \{v\}$ is the *closed neighborhood* of v. We omit subscripts when there is no ambiguity. The *degree* of a vertex v is $d(v) = |N(v)|$. An ordering $\alpha = (v_1, v_2, \ldots, v_n)$ of the vertices of a graph G is called a *non-increasing degree ordering* of G if $d(v_1) \geq d(v_2) \geq \cdots \geq d(v_n)$. A vertex is called *isolated* if its degree is 0. If $N[v] = V$, then we say that v is a *universal vertex* of G. A *path* in G is a sequence of distinct vertices $P = u_1u_2 \cdots u_p$, where u_iu_{i+1} is an edge of G for every $i = 1, \ldots, p-1$. We say that P is a path *between* u_1 and u_p, which are called the *end vertices* of P. If u_1u_p is an edge as well we obtain a *cycle*. A *forest* is a graph without cycles, and a *tree* is a connected forest. A vertex in a tree is called a *leaf* if it has degree 1. A *rooted tree* is a tree with a distinguished vertex called the *root*.

A graph is *connected* if there is a path between every pair of vertices. A maximal connected subgraph of a graph is called a *connected component*. A connected component of a graph is called *nontrivial* if it contains at least one edge. For any set $S \subseteq V$, we write $G[S]$ to denote the subgraph of G *induced* by S. We write $G - v$ to denote the graph $G[V \setminus \{v\}]$. The set S is said to be *connected* if $G[S]$ is connected. We say that two disjoint sets $S, S' \subseteq V$ are *adjacent* if there exist vertices $s \in S$ and $s' \in S'$ that are adjacent. A subset $S \subseteq V$ is a *clique* if all vertices in S are pairwise adjacent, and S is an *independent set* if no two vertices of S are adjacent. An *isomorphism* from a graph G to a graph H is a bijection $\varphi : V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $\varphi(u)\varphi(v) \in E(H)$. We say that G is *isomorphic* to H if there exists an isomorphism from G to H. The Induced Subgraph Isomorphism problem is to decide, given two graphs G and H, whether G has an induced subgraph that is isomorphic to H. We say that two rooted trees T_1 and T_2 are isomorphic if there is an isomorphism from T_1 to T_2 that maps the root of T_1 to the root of T_2.

The *contraction* of edge uv in G removes u and v from G, and replaces them by a new vertex, which is made adjacent to precisely those vertices that were adjacent to at least one of the vertices u and v. Instead of speaking of the contraction of edge uv, we sometimes say that a vertex

u is *contracted onto* v if the new vertex resulting from the contraction is still called v. We write G/uv to denote the graph obtained from G by contracting the edge uv. We say that a graph G can be *contracted* to a graph H, or is *H-contractible*, if H is isomorphic to a graph that can be obtained from G by a sequence of edge contractions. Let $S \subseteq V(G)$ be a connected set. If we repeatedly contract edges in $G[S]$ until only one vertex of $G[S]$ remains, we say that we *contract S into a single vertex*. Let H be a graph with vertex set $\{h_1, \ldots, h_{|V(H)|}\}$. Saying that a graph G can be contracted to H is equivalent to saying that G has a so-called *H-witness structure* $\mathcal{W}$, which is a partition of $V(G)$ into *witness sets* $W(h_1), \ldots, W(h_{|V(H)|})$, such that each witness set induces a connected subgraph of G, and such that for every two vertices $h_i, h_j \in V(H)$, the corresponding witness sets $W(h_i)$ and $W(h_j)$ are adjacent in G if and only if h_i and h_j are adjacent in H. By contracting each of the witness sets into a single vertex, we obtain a graph which is isomorphic to H. See Figure 2 for an example that shows that, in general, an H-witness structure of G is not uniquely defined. For any subset $S \subseteq V(H)$, we write $W(S)$ to denote the set of vertices of G that are contained in a witness set $W(v)$ for some $v \in S$, i.e., $W(S) = \cup_{v \in S} W(v)$.

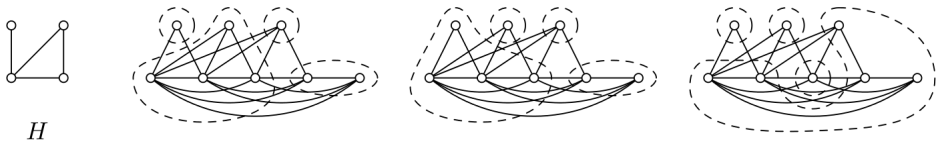


Figure 2: Three different H-witness structures of a threshold graph.

Cographs are the graphs that do not contain a path on four vertices as an induced subgraph. *Interval graphs* are the intersection graphs of intervals of a line, and they form a subclass of chordal graphs. *Chordal graphs* are the graphs without induced cycles of length more than 3.

Trivially perfect graphs have various characterizations [4, 14, 15, 28]. For our purposes, it is convenient to use the following characterization as a definition. A graph G is *trivially perfect* if and only if each connected induced subgraph of G contains a universal vertex [26, 27]. Let $\alpha = (v_1, v_2, \ldots, v_n)$ be an ordering of the vertices of a trivially perfect graph G. If α has the property that v_i is universal in a connected component of $G[\{v_i, , v_{i+1}, \ldots, v_n\}]$ for $i = 1, \ldots, n$, then α is called a *universal-in-a-component ordering (uco)*. A graph is trivially perfect if and only if it has a uco, and if and only if every non-increasing degree ordering is a uco [15, 28]. Consequently, for every edge uv in a trivially perfect graph, either $N[u] \subseteq N[v]$ or $N[v] \subseteq N[u]$ [28].

Every rooted tree T defines a connected trivially perfect graph, which is obtained by adding edges to T so that every path between the root and a leaf becomes a clique. In fact, all connected trivially perfect graphs can be created this way, and there is a bijection between rooted trees and connected trivially perfect graphs [28]. Given a connected trivially perfect graph G, a rooted tree T_G corresponding to G, which we call a *uco-tree* of G, can be obtained in the following way. If G is a single vertex, then T_G is this vertex. Otherwise, take a universal vertex v of G, make it the root of T_G, and delete it from G. In the remaining graph, for each connected component G', build a uco-tree $T_{G'}$ of G' recursively and make v the parent of the root of $T_{G'}$. All rooted trees that can be obtained from a connected trivially perfect graph in this way are isomorphic, and hence T_G is unique for every connected trivially perfect graph G. If G is disconnected, then it has a *uco-forest*, which is the disjoint union of the uco-trees of the connected components of G.

A graph G is a *split graph* if its vertex set can be partitioned into a clique C and an independent set I, where (C, I) is called a *split partition* of G. If C is not a maximum clique, then there is a vertex $v \in I$ that is adjacent to every vertex of C. In this case, $C' = C \cup \{v\}$ is a maximum clique, and $(C', I \setminus \{v\})$ is also a split partition of G. In this paper, unless otherwise stated, we assume that the clique C of a split partition (C, I) is maximum. This implies that none of the vertices in I is adjacent to every vertex of C. Split graphs form a subclass of chordal graphs.

Threshold graphs constitute a subclass of both trivially perfect graphs and split graphs. Threshold graphs have several characterizations [4, 14, 22], and we use the following one as a definition. A graph G is a *threshold graph* if and only if it is a split graph and, for any split partition (C, I) of G, there is an ordering $(v_1, v_2, \ldots, v_k)$ of the vertices of C such that $N[v_1] \supseteq N[v_2] \supseteq \ldots \supseteq N[v_k]$, and there is an ordering $(u_1, u_2, \ldots, u_\ell)$ of the vertices of I such that $N(u_1) \subseteq N(u_2) \subseteq \ldots \subseteq N(u_\ell)$ [22]. In that case, $(v_1, v_2, \ldots, v_k, u_\ell, \ldots, u_2, u_1)$ is a non-increasing degree ordering, and hence a uco, of G. Every connected threshold graph has a universal vertex, e.g., vertex v_1 in the ordering given above. Since we assume the clique of any split partition to be maximum, a vertex of C of smallest degree, e.g., vertex v_k in the ordering given above, has no neighbors in I. If a threshold graph is disconnected, then it has at most one nontrivial connected component; all other connected components are isolated vertices.

Split graphs, trivially perfect graphs, and threshold graphs are *hereditary* graph classes, meaning that the property of belonging to each of these classes is closed under taking induced subgraphs. These graph

classes can be recognized in linear time; split partitions and uco-trees can also be obtained in linear time [4, 14, 15, 28].

3 Contractions and Induced Subgraph Isomorphisms of Trivially Perfect Graphs

In this section, we will give results on the computational complexity of Contractibility on trivially perfect graphs, corresponding to the first four rows of Table 1. The first theorem reveals the equivalence of the problems Contractibility and Induced Subgraph Isomorphism on the class of connected trivially perfect graphs.

Theorem 1 *For any two connected trivially perfect graphs G and H, the following three statements are equivalent:*

(i) G can be contracted to H;

(ii) G contains an induced subgraph isomorphic to H;

(iii) T_G can be contracted to T_H.

Proof: First we prove the equivalence between (i) and (ii). Suppose G is H-contractible, and let uv be one of the edges of G that were contracted to obtain a graph isomorphic to H. Since G is trivially perfect, we have either $N_G[u] \subseteq N_G[v]$ or $N_G[v] \subseteq N_G[u]$. Without loss of generality, assume that $N_G[u] \subseteq N_G[v]$. Then contracting edge uv in G is equivalent to deleting vertex u from G. We can repeat this argument for every edge that was contracted, and conclude that G has an induced subgraph isomorphic to H.

For the opposite direction, suppose G' is an induced subgraph of G isomorphic to H. Let x be a universal vertex of G. We claim that G has an induced subgraph G'' isomorphic to H such that G'' contains x. If G' already contains x, then we can take $G'' = G'$. Suppose $x \notin V(G')$. Since G' is a connected trivially perfect graph, it has a universal vertex x'. Since x is a universal vertex in G, we have $N_G(x') \subseteq N_G[x]$. Hence the graph $G'' = G[(V(G') \setminus \{x'\}) \cup \{x\}]$ is isomorphic to G', and is therefore also isomorphic to H. Now let $y \neq x$ be one of the vertices that has to be deleted from G to obtain its induced subgraph G'', i.e., $y \in V(G) \setminus V(G'')$. Since x is a universal vertex, we know that $N_G(y) \subseteq N_G[x]$. Then deleting vertex y from G is equivalent to contracting edge xy in G. Since $x \in V(G'')$, we can repeat this argument for every vertex of $V(G) \setminus V(G'')$, and conclude that G is H-contractible.

Next we prove the equivalence between (ii) and (iii). Suppose G contains an induced subgraph G' isomorphic to H, and let y be one of the vertices of G that has to be deleted to obtain G'. As argued above, we can assume that G' contains a universal vertex $x \neq y$ of G, which we can assume to be the root of T_G. This means in particular that $G - y$ is connected. Let z be the parent of y in T_G, and let T' be the tree obtained from T_G by contracting y onto z. This makes z the parent in T' of all children of y in T_G. Other than this, all parent-children relations are the same in T' as they were in T_G. Since z was already adjacent in G to all the vertices in the subtree of T_G rooted at y, we see that T' is indeed a uco-tree of $G-y$, and hence T' is isomorphic to T_{G-y}. Now we can repeat this argument for every vertex of $V(G) \setminus V(G')$, and conclude that T_G is T_H-contractible.

For the opposite direction, suppose T_G is T_H-contractible, and let yz be one of the edges of T_G that were contracted to obtain a tree isomorphic to T_H. Let $T' = T_G/yz$, and assume without loss of generality that z is the parent of y in T_G and that y is contracted onto z. Let G' be the trivially perfect graph having T' as its uco-tree. Note that a vertex $u \neq y$ belongs to the subtree rooted at a vertex $v \neq y$ in T' if and only if u belongs to the subtree of T_G rooted at v. Therefore, by the definition of a uco-tree, $uv \in E(G')$ if and only if $uv \in E(G)$ for every pair of vertices $u, v \in V(G) \setminus \{y\}$, and hence G' is isomorphic to $G - y$. Now we can repeat this argument for every edge of T_G that was contracted, and conclude that G has an induced subgraph isomorphic to H. □

We point out that Theorem 1 does not hold when the connectivity requirement on G and H is dropped. For example, a connected trivially perfect graph G can have many disconnected induced subgraphs, but cannot be contracted to any of them. However, we will see that our polynomial-time algorithms in Theorems 3, 4, 5 and 6 below also work when G or H (or both) are disconnected.

Theorem 1 immediately gives us the result mentioned in the third row of Table 1, since checking whether a *fixed* graph H appears as an induced subgraph of an input graph G can trivially be done in polynomial time. Since Matoušek and Thomas [23] implicitly proved Contractibility to be NP-complete on rooted trees, Theorem 1 also implies the following result.

Corollary 2 *Both* Contractibility *and* Induced Subgraph Isomorphism *are NP-complete on connected trivially perfect graphs.*

Proof: Let T_1 and T_2 be two rooted trees given as input to Contractibility. Let G be the trivially perfect graph having T_1 as its

uco-tree, and let H be the trivially perfect graph having T_2 as its uco-tree. By Theorem 1, G is H-contractible if and only if G has an induced subgraph isomorphic to H if and only if T_1 is contractible to T_2. The corollary now follows from the result by Matoušek and Thomas [23], stating that CONTRACTIBILITY is NP-complete on rooted trees. □

The results below show that both problems can be solved in polynomial time when G is a trivially perfect graph and H is a threshold graph, even if both G and H are disconnected. Observe that these results are tight in light of Corollary 2. The following lemma is used in the proof of Theorem 3 below.

Lemma 1 *A connected trivially perfect graph G is a threshold graph if and only if every vertex in T_G has at most one child that is not a leaf.*

Proof: Recall that every threshold graph is trivially perfect. Let G be a threshold graph, and assume for a contradiction that there is a vertex x in T_G with two children u and v such that both u and v have children. This means that u and v are not adjacent in G, $N_G(u) \not\subseteq N_G(v)$, and $N_G(v) \not\subseteq N_G(u)$, which contradicts the assumption that G is a threshold graph. For the other direction, assume that G is trivially perfect and that every vertex in T_G has at most one child that is not a leaf. Let $P = p_1 p_2 \dots p_n$ be the unique path in T_G consisting of all the vertices that have at least one child, where p_1 is the root of T_G. Observe that $\{p_1, p_2, \dots, p_n\}$ is a clique in G. Since every vertex x of T_G is adjacent in G to all the vertices in the subtree of T_G rooted at x, the vertices of P satisfy $N_G[p_1] \supseteq N_G[p_2] \supseteq \dots \supseteq N_G[p_n]$. The leaves of T_G form an independent set in G. A leaf of T_G is adjacent in G to exactly those vertices that are ancestors of it in T_G. Since the leaves of T_G are only adjacent to vertices of P, there is also an ordering of them such that their neighborhoods are ordered by the subset relation. By the definition of threshold graphs, we can conclude that G is threshold. □

Theorem 3 *Given a threshold graph G and an arbitrary graph H, it can be decided in linear time whether G can be contracted to H.*

Proof: Our algorithm works as follows. First we check if H has at most as many vertices and edges as G, and reject if not. Since threshold graphs are hereditary, G is H-contractible only if H is a threshold graph. We can check in linear time whether this is the case, and reject if not. Suppose H is a threshold graph. Since edge contractions preserve connectivity, we can immediately reject if G and H do not have the same number of

connected components. Suppose G and H have the same number of connected components. We trivially output "yes" if H contains no edges. Assume that both G and H contain at least one edge. Recall that any threshold graph contains at most one nontrivial connected component. Now the problem is equivalent to deciding whether the only nontrivial connected component of G can be contracted to the only nontrivial connected component of H. Hence for the rest of the proof we can assume G and H to be connected threshold graphs. By Theorem 1, our remaining task is equivalent to deciding whether the uco-tree T_G of G can be contracted to the uco-tree T_H of H.

We compute the uco-trees T_G and T_H in linear time. Since G is a threshold graph, we know by the proof of Lemma 1 that T_G has a unique path containing all the vertices that have at least one child. Let $S = s_1 s_2 \cdots s_g$ be this path, where s_1 is the root of T_G. Let $T = t_1 t_2 \cdots t_h$ be an analogous path in T_H. Let $l(v)$ be the number of leaves adjacent to a vertex v of T or S. We describe an algorithm that either finds a T_H-witness structure $\mathcal{W}$ of T_G, or concludes that T_G is not T_H-contractible. First we distribute the vertices of S over the sets $W(t_1), \ldots, W(t_h)$ according to the following greedy procedure. Initially, we set $W(x) = \emptyset$ for each vertex x of T_H.

$j = 1$;
for $i = 1$ **to** h **do**
    $\ell = 0$;
    repeat
        $W(t_i) = W(t_i) \cup \{s_j\}$;
        $\ell = \ell + l(s_j)$;
        $j = j + 1$;
    until ($\ell \geq l(t_i)$ **or** $j > g$);
    if $j > g$ **and** ($\ell < l(t_i)$ **or** $i < h$) **then**
        stop;
end for;
if $j \leq g$ **then**
    $W(t_h) = W(t_h) \cup \{s_j, \ldots, s_g\}$;

If this procedure runs until the end without being terminated by the **stop** command, then for $i = 1, \ldots, h$ we know that T_G has at least $l(t_i)$ leaves adjacent to vertices that we placed in $W(t_i)$. For each leaf t of T_H adjacent to t_i, we take a different leaf s of T_G adjacent to a vertex of $W(t_i)$, and we let $W(t) = \{s\}$. If T_G has any leaves that are adjacent to $W(t_i)$ but have not been assigned to witness sets of cardinality 1 like this, then we add all those leaves to $W(t_i)$. We repeat this for each i.

Since the above procedure places each vertex of S in a witness set, this partitions all vertices of T_G into witness sets.

We first remark that the number of witness sets adjacent to each $W(t_i)$ is equal to the degree of t_i, and therefore $\mathcal{W}$ is a T_H-witness structure of G. Hence, if the above procedure runs until the end without being terminated by the **stop** command, then it produces a T_H-witness structure of T_G, and hence T_G is T_H-contractible.

Now we prove that if the procedure is terminated by the **stop** command before reaching the end of the **for**-loop, then we can conclude that T_G is not T_H-contractible. Assume for a contradiction that T_G is T_H-contractible, but that the procedure is terminated by the **stop** command. Let $W(t_1), \ldots, W(t_p)$ be the witness sets that the procedure generated before it terminated. Let $\mathcal{W}'$ be a correct T_H-witness structure of T_G. From the proof of Theorem 1 it is clear that we may assume that $s_1 \in W'(t_1)$. Since every $t_i \in T$ with $i \geq 2$ has at least 2 neighbors in T_H, and all the vertices in T_G that are not in S have degree 1, every witness set $W'(t_i)$ contains at least one vertex of S. The connectivity of the witness sets implies that $(W'(t_1) \cup \ldots \cup W'(t_h)) \cap S = S'$, where $S' = \{s_1, \ldots, s_j\}$ for some integer j. Moreover, the sets $W'(t_1), \ldots, W'(t_h)$ partition S' into exactly h subpaths, and each of these witness sets contains consecutive vertices of S'. Now let k be the smallest integer such that $W'(t_k)$ differs from $W(t_k)$; note that $k \leq p$, but not necessarily $k = p$. Since k is chosen to be smallest, the vertex of S' with the smallest index in $W'(t_k)$ is the same as the vertex of S' with smallest index in $W(t_k)$. Observe that the number of leaves of T_G adjacent to the vertices of $W'(t_k)$ is at least $l(t_k)$. The **repeat**-loop for building $W(t_k)$ stops as soon as this number is reached, and hence $W(t_k)$ does not contain more vertices of S' than $W'(t_k)$. Since the two sets are different, we conclude that $W(t_k)$ contains fewer vertices of S' than $W'(t_k)$, meaning that $W(t_k) \subset W'(t_k)$. Consequently, k was not the step at which the above procedure stopped. Furthermore, the vertex of S' with the smallest index in $W(t_{k+1})$ has a smaller index than the vertex of S' with the smallest index in $W'(t_{k+1})$, and by the same arguments, the vertex of S' with the largest index in $W(t_{k+1})$ has no larger index than the vertex of S' with the largest index in $W'(t_{k+1})$. Now we can repeat the same arguments to conclude that the vertex of S' with the largest index in $W(t_i)$ has no larger index than the vertex of S' with the largest index than $W'(t_i)$, for $i = k+2, \ldots, p$, which contradicts the assumption that the procedure terminated after generating the set $W(t_p)$.

All the described steps can clearly be completed within a running time of $O(|V(G)| + |E(G)|)$ if G and H are given by their adjacency lists.

If G and H are already recognized as threshold graphs before testing contractibility, and if they are provided to us in a compact representation, like a non-increasing degree order, then our running time becomes $O(|V(G)|)$. □

Theorem 4 *Given a trivially perfect graph G and a threshold graph H, it can be decided in polynomial time whether G can be contracted to H.*

Proof: If G has less vertices or edges than H, then we return a negative answer. If G or H is disconnected, we first check whether G and H have the same number of connected components. If not, then we return a negative answer, since edge contractions preserve connectivity. Otherwise, for every connected component G' of G and the unique nontrivial connected component H' of H, we check if G' is H'-contractible. By Theorem 1, this is equivalent to testing whether $T_{G'}$ can be contracted to $T_{H'}$. The total running time is no worse than $O(|V(G)|)$ times the running time of checking contractibility on a pair of connected components. Hence for the rest of the proof we assume that input graphs G and H are connected.

We compute the uco-trees T_G and T_H in linear time. Let S be any path in T_G between the root and the parent of a leaf. We define $C(S)$ to be the graph obtained by contracting every edge of T_G, apart from the edges that have both endpoints in S or that are incident to a leaf. By Theorem 1, $C(S)$ is the uco-tree of an induced subgraph G_S of G, and by Lemma 1, G_S is a threshold graph. Note also that $C(S)$ has as many leaves as G. We claim that G is H-contractible if and only if there is a path S in T_G such that $C(S)$ is T_H-contractible. Clearly, if such a path S exists, then T_G is T_H-contractible, and hence G is H-contractible by Theorem 1.

We now prove that if G is H-contractible, then such a path S exists in T_G. Assume that G is H-contractible. Then we know by Theorem 1 and its proof that G has an induced subgraph G' isomorphic to H such that G' contains the root of T_G. Hence we can assume that T_G and $T_{G'}$ have the same root. Since G' is a threshold graph, by Lemma 1, $T_{G'}$ has the property that there is a unique maximal path from the root every vertex of which has at least one child in $T_{G'}$. Let $T = t_1 t_2 \cdots t_h$ be such a path in $T_{G'}$. Hence t_1 is the root of $T_{G'}$, and t_h is the lowest vertex that is not a leaf. Let $\mathcal{W}$ be such a $T_{G'}$-witness structure of T_G. Using similar arguments as the ones in the proof of Theorem 3, we may assume that $W(t_1)$ contains the root of T_G, and that there exists a path S in T_G from the root to the parent of a leaf such that $W(t_1), \ldots, W(t_h)$ partition S into exactly h subpaths. For each $i \in \{1, \ldots, h\}$, let T_i be the subgraph of

T_G obtained by deleting the vertices belonging to $W(t_j)$ for all $j \neq i$. The connected component of T_i containing $W(t_i)$ contains at least as many leaves of T_G as the number of leaves which are neighbors of t_i in $T_{G'}$. Hence $C(S)$ is a graph that is $T_{G'}$-contractible. Since G' is isomorphic to H, $T_{G'}$ is isomorphic to T_H, and consequently S is exactly the path whose existence in T_G we wanted to prove.

The algorithm is now clear from the above discussion. For each distinct maximal path S of T_G from the root containing only vertices that have at least one child, we check whether $C(S)$ is contractible to T_H using the linear-time procedure described in the proof of Theorem 3. Since the number of distinct paths S is $O(|V(G)|)$, the total running time is polynomial. □

By Theorem 1, Induced Subgraph Isomorphism is equivalent to Contractibility on connected trivially perfect graphs. Hence the only difference between the proofs of the following results and those of the two previous theorems is in the connectivity arguments.

Theorem 5 *Given a trivially perfect graph G and a threshold graph H, it can be decided in polynomial time whether G contains an induced subgraph isomorphic to H.*

Proof: Let G be a trivially perfect graph and let H be a threshold graph. We construct a graph G' from G by adding a new vertex x and making it adjacent to all vertices of G. Note that G' is a connected trivially perfect graph. Let H' be the connected threshold graph obtained from H by adding a new vertex y and making it adjacent to all vertices of H. We claim that G has an induced subgraph isomorphic to H if and only if G' has an induced subgraph isomorphic to H'. Assume that G' has an induced subgraph G'' isomorphic to H'. By the arguments used in the proof of Theorem 1, we can assume that G'' contains x. Consequently, $G'' - x$ is an induced subgraph of G. Since $G'' - x$ is isomorphic to H, this direction of the claim follows. For the other direction, assume that there exists a subset $U \subseteq V(G)$ such that $G[U]$ is isomorphic to H. Then the subgraph of G' induced by $U \cup \{x\}$ is isomorphic to H'. Hence, in order to prove Theorem 5, it suffices to show that we can decide in polynomial time whether a connected trivially perfect graph G' can be contracted to a connected threshold graph H'. This follows from Theorems 1 and 4. □

Theorem 6 *Given a threshold graph G and an arbitrary graph H, it can be decided in linear time whether G has an induced subgraph isomorphic to H.*

Proof: Since threshold graphs are trivially perfect, we can use the same arguments as in the proof of Theorem 5 to conclude that it is enough to consider connected input graphs. Now the result follows from Theorems 1 and 3. □

4 Contracting Split Graphs

In the previous section, we showed that it can be decided in linear time whether a threshold graph G can be contracted to an arbitrary graph H. The next theorem shows that this result is not likely to be extendable to split graphs. A *hypergraph* F is a pair $(Q, \mathcal{S})$ consisting of a set $Q = \{q_1, \ldots, q_k\}$, called the *vertices* of F, and a set $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ of nonempty subsets of Q, called the *hyperedges* of F. A 2-*coloring* of a hypergraph $F = (Q, \mathcal{S})$ is a partition (Q_1, Q_2) of Q such that $Q_1 \cap S_j \neq \emptyset$ and $Q_2 \cap S_j \neq \emptyset$ for $j = 1, \ldots, \ell$.

Theorem 7 Contractibility *is NP-complete on input pairs* (G, H) *where* G *is a connected split graph and* H *is a connected threshold graph.*

Proof: We use a reduction from Hypergraph 2-Colorability, which is the problem of deciding whether a given hypergraph has a 2-coloring. This problem, also known as Set Splitting, is NP-complete [21]. The problem remains NP-complete when restricted to hypergraphs in which every vertex is contained in at least two hyperedges.

Let $F = (Q, \mathcal{S})$ be a hypergraph with $Q = \{q_1, \ldots, q_k\}$ and $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ such that every vertex of Q appears in at least two hyperedges. We construct a split graph G as follows. We start with a clique $A = \{a_1, \ldots, a_k\}$, where the vertex $a_i \in A$ corresponds to the vertex $q_i \in Q$ for $i = 1, \ldots, k$. We add an independent set $B = \{b_1, \ldots, b_\ell\}$, where the vertex $b_i \in B$ corresponds to the hyperedge $S_i \in \mathcal{S}$ for $i = 1, \ldots, \ell$. Finally, for $i = 1, \ldots, k$ and $j = 1, \ldots, \ell$, we add an edge between a_i and b_j in G if and only if $q_i \in S_j$. We also construct a threshold graph H from a single edge x_1x_2 by adding an independent set $Y = \{y_1, \ldots, y_\ell\}$ on ℓ vertices, and making each vertex of Y adjacent to both x_1 and x_2. We claim that G can be contracted to H if and only if F has a 2-coloring.

Suppose F has a 2-coloring, and let (Q_1, Q_2) be a 2-coloring of F. Let (A_1, A_2) be the partition of A corresponding to this 2-coloring of F. Note that A_1 and A_2 both form a connected set in G, since the vertices of A form a clique in G. We contract A_1 into a single vertex p_1, and we contract A_2 into a single vertex p_2. Let G' denote the resulting graph. Since (Q_1, Q_2) is a 2-coloring of F, every vertex in B is adjacent to at

least one vertex of A_1 and at least one vertex of A_2 in the graph G. As a result, every vertex in B is adjacent to both p_1 and p_2 in G'. Hence G' is isomorphic to H, which means that G can be contracted to H.

Now suppose G can be contracted to H, and let $\mathcal{W}$ be an H-witness structure of G. Since we assumed that every vertex of F appears in at least two hyperedges, every vertex in A has at least two neighbors in B. This means that B is the only independent set of size ℓ in G. Since Y is an independent set of size ℓ in H, the witness sets $W(y_1), \ldots, W(y_\ell)$ each must contain exactly one vertex of B. In fact, since every vertex of A has at least two neighbors in B, we have $W(Y) = B$. This means that the two witness sets $W(x_1)$ and $W(x_2)$ form a partition of the vertices of A. By the definition of an H-witness structure and the construction of H, each witness set $W(y_i)$ is adjacent to both $W(x_1)$ and $W(x_2)$. Hence the partition $(W(x_1), W(x_2))$ of A corresponds to a 2-coloring of F. □

Although Theorem 7 shows that the problem of deciding whether a split graph G can be contracted to a split graph H is NP-complete when both G and H are given as input, we will show in the remainder of this section that the problem can be solved in polynomial time when H is fixed.

Definition 1 *Let G and H be two split graphs with split partitions (C_G, I_G) and (C_H, I_H), respectively. A set $U \subseteq I_G$ with $|U| = |I_H|$ is called* H-compatible *if G has an H-witness structure $\mathcal{W}$ such that $W(I_H) = U$.*

Lemma 2 *Let G and H be two split graphs. Then G is H-contractible if and only if G contains an H-compatible set.*

Proof: If G has an H-compatible set, then G is H-contractible by Definition 1. For the reverse direction, assume that G is H-contractible, and let $\mathcal{W}$ be an H-witness structure of G. If I_H is empty, then $U = \emptyset$ is an H-compatible set of G by Definition 1, since the H-witness structure $\mathcal{W}$ satisfies $W(I_H) = U = \emptyset$. Suppose I_H is not empty. Since I_H is an independent set in H, there can be at most one vertex $v \in I_H$ such that $W(v)$ contains a vertex of C_G. Note that this implies that G is not H-contractible if $|I_G| \leq |I_H| - 1$. Suppose there is a witness set $W(v)$ such that $W(v) \cap C_G \neq \emptyset$. Then for each $v' \in I_H \setminus \{v\}$, the witness set $W(v')$ contains only vertices of I_G, i.e., $W(I_H \setminus \{v\}) \subseteq I_G$. Since I_G is an independent set in G and every witness set is connected, $|W(v')| = 1$ for every $v' \in I_H \setminus \{v\}$. Recall that C_H is assumed to be a maximum clique of H. Hence there is a vertex x of C_H that is not adjacent to v in H, and therefore witness set $W(x)$ is not adjacent to $W(v)$ in G. Since $W(v)$ contains

at least one vertex of C_G and is not adjacent to $W(x)$, $W(x)$ only contains vertices of I_G. Since I_G is an independent set and $W(x)$ is connected, we must have $W(x) = \{a\}$ for some vertex $a \in I_G$. This implies that $W(v)$ is adjacent to witness set $W(x')$ for every $x' \in C_H \setminus \{x\}$. Moreover, since for every $v' \in I_H \setminus \{v\}$ the witness set $W(v')$ consists of a single vertex from I_G, $W(x)$ is not adjacent to $W(v')$ for any $v' \in I_H \setminus \{v\}$. Therefore, we can define another H-witness structure $\mathcal{W}'$ of G by setting $W'(v) = W(x)$, $W'(x) = W(v)$, and $W'(y) = W(y)$ for every $y \in V(H) \setminus \{v, x\}$. Now $\mathcal{W}'$ has the property that $|W'(y)| = 1$ for every vertex $y \in I_H$. Consequently, $U = W'(I_H)$ is an H-compatible set of G by Definition 1. □

If U is an H-compatible set of G, then, by Definition 1, G has an H-witness structure $\mathcal{W}$ such that $W(I_H) = U$. The next technical lemma shows that each of the witness sets of $\mathcal{W}$ contains a small subset, bounded in size by a function of $|V(H)|$ only, such that the collection of these subsets provide all the necessary adjacencies between the witness sets of $\mathcal{W}$.

Lemma 3 *Let G and H be two connected split graphs with split partitions (C_G, I_G) and (C_H, I_H), respectively. Let $C_H = \{x_1, \ldots, x_k\}$. A set $U \subseteq I_G$ with $|U| = |I_H|$ is H-compatible if and only if there exists a collection $\mathcal{M}$ of pairwise disjoint subsets $M(x_1), \ldots, M(x_k)$ of $V(G) \setminus U$ satisfying the following properties:*

- *(i) at most one set of $\mathcal{M}$ contains a vertex of I_G, and such a set has cardinality 1 if it exists;*
- *(ii) for every subset $X \subseteq U$, $M(x_i)$ contains at most two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U = X$, for $i = 1, \ldots, k$;*
- *(iii) $\bigcup_{i=1}^{k} |M(x_i)| \leq |C_H| \cdot 2^{|I_H|+1}$;*
- *(iv) for every $v \in V(G) \setminus (U \cup \bigcup_{i=1}^{k} M(x_i))$, there is a set in $\mathcal{M}$ that is adjacent to every vertex in $N_G(v) \cap U$;*
- *(v) the graph $G' = G[U \cup \bigcup_{i=1}^{k} M(x_i)]$ has an H-witness structure $\mathcal{W}'$ such that $W'(I_H) = U$ and $W'(x_i) = M(x_i)$ for $i = 1, \ldots, k$.*

Proof: Let U be a subset of I_G of cardinality $|I_H|$. Suppose there exists a collection $\mathcal{M} = \{M(x_1), \ldots, M(x_k)\}$ that has properties (i)–(v). Let $M = \bigcup_{i=1}^{k} M(x_i)$, and let $G' = G[U \cup M]$. By property (v), there exists an H-witness structure $\mathcal{W}'$ of G' such that $W'(I_H) = U$ and $W'(x_i) = M(x_i)$ for $i = 1, \ldots, k$. We will show that $\mathcal{W}'$ can be extended to an H-witness structure $\mathcal{W}$ of G with $W(I_H) = U$. Let $v \in C_G \setminus M$. By property (iv),

there is a set $W'(x_i) \in \mathcal{W}'$ that is adjacent to every vertex of $N_G(v) \cap U$. Add v to $W'(x_i)$. Note that adding v to $W'(x_i)$ does not change the adjacencies between $W'(x_i)$ and the other witness sets of $\mathcal{W}'$. Repeat this until all vertices of $C_G \setminus M$ have been added to sets of $\mathcal{W}'$. Let $w \in I_G$. Since every vertex of C_G now belongs to a set of $\mathcal{W}'$, there exists a set $W'(x_j) \in \mathcal{W}'$ that is adjacent to w. Add w to $W'(x_j)$. It is clear that adding w to $W'(x_j)$ does not change the adjacencies between $W'(x_j)$ and the other witness sets of $\mathcal{W}'$. Repeat this until all vertices of $I_G \setminus U$ have been added to sets of $\mathcal{W}'$. We end up with an H-witness structure $\mathcal{W}$ of G with $W(I_H) = U$, which means that U is H-compatible by Definition 1.

For the reverse direction, suppose that U is an H-compatible set of G. Then, by definition, G has an H-witness structure $\mathcal{W}$ such that $W(I_H) = U$. Suppose there exist a vertex $x_i \in C_H$ whose witness set $W(x_i)$ contains only vertices of I_G. Note that this means that x_i is not adjacent to any vertex in I_H, as no vertex of I_G has a neighbor in U. Since every witness set is a connected set, $W(x_i) = \{p\}$ for some vertex $p \in I_G$. Suppose there is another set $W(x_j)$ that contains only vertices of I_G. Then $W(x_j) = \{q\}$ for some $q \in I_G \setminus \{p\}$. Since x_i and x_j are adjacent in H, the witness sets $W(x_i)$ and $W(x_j)$ must be adjacent in G. This contradicts the fact that p and q, both belonging to the independent set I_G, are not adjacent. This implies that there is at most one vertex $x_i \in C_H$ such that $W(x_i) = \{p\}$ for some $p \in I_G$. Moreover, if such a witness set exists, then p has at least one neighbor in the witness set $W(x_j)$ for every $x_j \in C_H$, since $\mathcal{W}$ is an H-witness structure and C_H is a clique in H.

We now show how to construct the collection $\mathcal{M}$ from $\mathcal{W}$. For $i = 1, \ldots, k$, the set $M(x_i)$ is a subset of the witness set $W(x_i)$, and $M(x_i)$ can be obtained from $W(x_i)$ as follows. We first partition the vertices of $W(x_i)$ into sets in such a way, that two vertices a and b of $W(x_i)$ belong to the same partition set if and only if they are adjacent to the same vertices in U, i.e., if $N_G(a) \cap U = N_G(b) \cap U$. Let $S_i \subseteq W(x_i)$ be the partition set whose vertices have no neighbor in U. From each non-empty partition set other than S_i, we arbitrarily choose one vertex and add it to $M(x_i)$. If $S_i \neq W(x_i)$, then no vertex of S_i is added to $M(x_i)$. If $S_i = W(x_i)$ and S_i contains at least one vertex of C_G, then we arbitrarily choose one of the vertices of $S_i \cap C_G$ and add it to $M(x_i)$. If $S_i = W(x_i)$ and S_i contains no vertices of C_G but contains a vertex of I_G, then we add that vertex to $M(x_i)$; recall that in this case S_i contains exactly one vertex, and that this case occurs at most once. After we have generated all the sets $M(x_i)$ this way, we check if there is a set $M(x_j) = \{p\}$ for some $p \in I_G$. If so, then we check, for every $x_i \in C_H \setminus \{x_j\}$, whether the set $M(x_i)$

contains at least one neighbor of p. If not, then we arbitrarily choose a neighbor p' of p in $W(x_i)$ and add it to $M(x_i)$. As we argued before, such a neighbor p' always exists. Note that adding p' to $M(x_i)$ does not change the adjacencies between $M(x_i)$ and U, since $M(x_i)$ already contained one vertex from every non-empty partition set of $W(x_i)$.

Let $\mathcal{M}$ be the collection of sets $M(x_i)$ that are obtained this way from the witness sets $W(x_i)$, for every $x_i \in C_H$. For every $x_i \in C_H$, every vertex of $W(x_i) \setminus S_i$ belongs to C_G, since no vertex of I_G has a neighbor in U. The only time a vertex of I_G is added to a set $M(x_i)$ is when $S_i = W(x_i)$ and $W(x_i)$ does not contain a vertex of C_G. As we argued above, this situation occurs at most once, so $\mathcal{M}$ satisfies property (i). For every witness set $W(x_i)$, there are at most $2^{|I_H|}$ non-empty partition sets, since U is H-compatible and thus has cardinality $|I_H|$. The set $M(x_i)$ contains one vertex from each non-empty partition set, and possibly one extra vertex a which is adjacent to the only set in $\mathcal{M}$ of the form $M(x_j) = \{p\}$ for some $p \in I_G$. If such a vertex a exists, then this is the only vertex of $M(x_i)$ for which there exists another vertex $b \in M(x_i)$ with $N_G(a) \cap U = N_G(b) \cap U$. Hence every set $M(x_i)$ contains at most two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U$, for $i = 1, \ldots, k$, and therefore certainly satisfies property (ii). The reason we write "for every subset $X \subseteq U$" instead of "for at most one subset $X \subseteq U$" in property (ii) will become clear from the description of the algorithm in Case 2 in the proof of Lemma 4. The number of non-empty partition sets is at most $2^{|I_H|}$, so $|M(x_i)| \leq 2^{|I_H|} + 1 < 2^{|I_H|+1}$. This, together with the fact that $k = |C_H|$, implies property (iii). Again, the reason for not formulating property (iii) in the strongest possible way will become clear in the proof of Lemma 4. Let $v \in V(G) \setminus (U \cup \bigcup_{i=1}^{k} M(x_i))$. Since $v \notin U$, v belongs to a witness set $W(x_i)$ for some $x_i \in C_H$. Consider the set $M(x_i)$. By construction, there exists a vertex $w \in M(x_i)$ such that $N_G(v) \cap U = N_G(w) \cap U$, as otherwise v would have been added to $M(x_i)$. Hence $M(x_i)$ is adjacent to all vertices in $N_G(v) \cap U$, and property (iv) holds.

It remains to show $\mathcal{M}$ satisfies property (v). For every $x_i \in C_H$, the set $M(x_i)$ is adjacent to exactly the same vertices in U as the set $W(x_i)$, since $M(x_i)$ contains a vertex from every partition class of $W(x_i)$. If every set in $\mathcal{M}$ contains at least one vertex of C_G, then the fact that C_G is a clique in G implies that the sets of $\mathcal{M}$ are pairwise adjacent. Hence property (v) holds in this case. Suppose $\mathcal{M}$ contains a set of the form $M(x_j) = \{p\}$ for some $p \in I_G$. Since $\mathcal{M}$ satisfies property (i), every set in $\mathcal{M} \setminus M(x_j)$ contains only vertices from C_G, which means that those sets are pairwise adjacent. The last step in the construction of $\mathcal{M}$ ensures

that p is adjacent to every set in $\mathcal{M} \setminus M(x_j)$. Hence property (v) also holds in this case. □

We call the collection $\mathcal{M}$ in Lemma 3 an *essential collection* for U, and the sets $M(x_i)$ are called *essential sets*. The fact that the total size of an essential collection does not depend on the size of G plays a crucial role in the proof of the following lemma.

Lemma 4 *Let G and H be two split graphs with split partitions (C_G, I_G) and (C_H, I_H), respectively. Given a set $U \subseteq I_G$ with $|U| = |I_H|$, it can be decided in $f(|V(H)|) \cdot |V(G)|^3$ time whether U is H-compatible, where the function f depends only on H and not on G.*

Proof: Let U be a subset of I_G with $|U| = |I_H|$, and let $C_H = \{x_1, \ldots, x_k\}$. Throughout the proof, we use k to represent the number of vertices in C_H. We present an algorithm that checks whether or not there exists an essential collection for U. By Lemma 3, U is H-compatible if and only if such a collection exists. We distinguish two cases, depending on whether or not every vertex of C_H has at least one neighbor in I_H.

Case 1. Every vertex of C_H has at least one neighbor in I_H.

For every subset $X \subseteq U$, we define the set $Z_X = \{v \in V(G) \setminus U \mid N_G(v) \cap U = X\}$. Note that there are at most $2^{|U|}$ non-empty sets Z_X, and that these sets form a partition of $V(G) \setminus U$. Let $\mathcal{Z} = \{Z_X \mid X \subseteq U\}$ be the collection of these sets Z_X. Let $\mathcal{A}$ be the power set of $\mathcal{Z}$, i.e., $\mathcal{A}$ is the set consisting of all possible subsets of $\mathcal{Z}$. For every element $A \in \mathcal{A}$, we have $A = \{Z_{X_1}, \ldots, Z_{X_\ell}\}$ for some $1 \leq \ell \leq 2^{|U|}$, where $X_i \subseteq U$ for $i = 1, \ldots, \ell$ and $X_i \neq X_j$ whenever $i \neq j$. Finally, let $\mathcal{B}$ be the set of all ordered k-tuples of elements in $\mathcal{A}$, where elements of $\mathcal{A}$ may appear more than once in an element $B \in \mathcal{B}$. For any element $B \in \mathcal{B}$, we have $B = (A_1, A_2, \ldots, A_k)$, where $A_i \in \mathcal{A}$ for $i = 1, \ldots, k$.

For every $B = (A_1, A_2, \ldots, A_k) \in \mathcal{B}$, we generate a "candidate" essential set $M(x_i)$ for every vertex $x_i \in C_H$ as follows. At the start, all the vertices of C_G are unmarked, and all the vertices of $I_G \setminus U$ are marked. Of every set in A_1 that contains at least one unmarked vertex, we add one unmarked vertex to $M(x_1)$. We mark all the vertices that are added to $M(x_1)$. We then generate a candidate essential set $M(x_2)$ as before, adding an unmarked vertex from every set in A_2 that contains such a vertex to $M(x_2)$, and marking all the vertices added to $M(x_2)$. After we have generated a candidate essential set $M(x_i)$ for every vertex $x_i \in C_H$ in the way described, we define $M = \bigcup_{i=1}^{k} M(x_i)$, i.e., M is the set of marked vertices of C_G. Let $\mathcal{M}$ denote the collection of all candidate essential sets $M(x_i)$. Note that the sets of $\mathcal{M}$ are pairwise disjoint subsets

of C_G. It is clear that, by construction, $\mathcal{M}$ satisfies properties (i), (ii), and (iii) of Lemma 3.

We now check whether $\mathcal{M}$ satisfies properties (iv) and (v). In order to check property (iv), we determine for every vertex $v \in V(G) \setminus (U \cup M)$ whether $\mathcal{M}$ contains a candidate essential set that is adjacent to every vertex in $N_G(v) \cap U$. $\mathcal{M}$ satisfies property (iv) if and only if such a set exists for every vertex of $V(G) \setminus (U \cup M)$. In order to check property (v), we first delete all the vertices in $V(G) \setminus (U \cup M)$, and then contract each of the candidate essential sets $M(x_i)$ into a single vertex. $\mathcal{M}$ satisfies property (v) if and only if the obtained graph is isomorphic to H. If $\mathcal{M}$ satisfies properties (iv) and (v), then $\mathcal{M}$ is an essential collection for U, and the algorithm concludes that U is H-compatible. If $\mathcal{M}$ does not satisfy properties (iv) and (v), then we unmark all vertices of C_G (the vertices of $I_G \setminus U$ remain marked) and repeat the procedure on the next element of $\mathcal{B}$. If we have processed all elements of $\mathcal{B}$ without finding an essential collection for U, then we conclude that U is not H-compatible due to Lemma 3.

Before we consider Case 2 below, we first prove why the algorithm for Case 1 is correct. If the algorithm finds a collection $\mathcal{M}$ that satisfies properties (i)–(v), then $\mathcal{M}$ is an essential collection for U by definition. Hence, by Lemma 3, the algorithm correctly concludes that U is H-compatible in this case. It remains to prove that if U is H-compatible, then our algorithm will find an essential set $\mathcal{M}$ for U.

Suppose U is H-compatible. Then, by definition, G has an H-witness structure $\mathcal{W}'$ such that $W'(I_H) = U$. Let $\mathcal{M}'$ be an essential collection for U, obtained from $\mathcal{W}'$ in the way described in the proof of Lemma 3. Since every vertex of C_H has at least one neighbor in I_H, every set $M'(x_i) \in \mathcal{M}'$ satisfies the following two properties by construction: $M'(x_i)$ contains no vertex of I_G, and $M'(x_i)$ does not contain two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U$, i.e., $M'(x_i)$ contains at most one vertex from every set Z_X, for every subset $X \subseteq U$. Note that the sets of $\mathcal{M}'$ are pairwise disjoint, which means that, for every set Z_X, the number of vertices in Z_X is at least as big as the number of sets of $\mathcal{M}'$ that contain a vertex of Z_X. Consider the set $M'(x_1)$. Let $A_1 = \{Z_{X_1}, \ldots, Z_{X_\ell}\}$ be the collection of sets in $\mathcal{Z}$ such that $M'(x_1) \cap Z_{X_i} \neq \emptyset$ for $i = 1, \ldots, \ell$. Let $A_2, \ldots, A_k$ be defined similarly for the sets $M'(x_2), \ldots, M'(x_k)$, respectively. Let $B = (A_1, \ldots, A_k)$. Since our algorithm processes every element in $\mathcal{B}$ if necessary, it will consider B at some stage, unless it found an essential collection for U before and correctly concluded that U is H-compatible. When processing B, the algorithm will create candidate essential sets $M(x_1), \ldots, M(x_k)$ such that, for $i = 1, \ldots, k$, the set $M(x_i)$ contains a

vertex from exactly those sets in $\mathcal{Z}$ that $M'(x_i)$ contains a vertex from. Just like the sets $M'(x_i)$, the sets $M(x_i)$ are pairwise disjoint subsets of C_G. Since $N_G(a) \cap U = N_G(b) \cap U$ for every two vertices a, b that belong to the same set of $\mathcal{Z}$, the collection $\mathcal{M} = \{M(x_1), \ldots, M(x_k)\}$ satisfies properties (i)–(v) of Lemma 3, and thus is an essential collection for U.

Case 2. At least one vertex of C_H has no neighbor in I_H.

Let S be the set of vertices of C_H that do not have any neighbors in I_H. Assume, without loss of generality, that $x_k \in S$. We first run the algorithm for Case 1 on U. If we find an essential set for U this way, then we conclude that U is H-compatible. Suppose we do not find an essential set for U using the algorithm for Case 1. We then run the following algorithm for every vertex $p \in I_G \setminus U$.

We first define a candidate essential set for x_k by setting $M(x_k) = \{p\}$. For every subset $X \subseteq U \cup \{p\}$, we define the set $Z_X = \{v \in V(G) \setminus (U \cup \{p\}) \mid N_G(v) \cap (U \cup \{p\}) = X\}$. Let $\mathcal{Z} = \{Z_X \mid X \subseteq (U \cup \{p\})\}$ be the collection of these sets Z_X, and let $\mathcal{A}$ be the power set of $\mathcal{Z}$. Let $\mathcal{B}$ be the set of all ordered $(k-1)$-tuples of elements in $\mathcal{A}$, where elements of $\mathcal{A}$ may appear more than once in an element $B \in \mathcal{B}$. Then, for every $B = (A_1, \ldots, A_{k-1}) \in \mathcal{B}$, we act as follows. We mark all the vertices of $I_G \setminus U$, and leave the vertices of C_G unmarked. In particular, p is marked. For every i from 1 to $k-1$, we generate a candidate essential set $M(x_i)$ as we did in Case 1: from every set in A_i that contains at least one unmarked vertex, we add one unmarked vertex to $M(x_i)$. We mark all the vertices that are added to $M(x_i)$.

Let $\mathcal{M}$ be the collection of candidate essential sets generated this way, and let $M = \bigcup_{i=1}^{k} M(x_i)$ be the set of all marked vertices, including p. Since we marked all the vertices of $I_G \setminus U$ at the start of the algorithm, only the candidate essential set $M(x_k) = \{p\}$ contains a vertex from I_G. Hence $\mathcal{M}$ satisfies property (i) of Lemma 3. Every set $M(x_i)$ contains at most one vertex from every set in $\mathcal{Z}$, and therefore never contains two vertices with exactly the same neighbors in $U \cup \{p\}$. It is however possible, for every $X \subseteq U$, that $M(x_i)$ contains two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U = X$, in which case exactly one of these two vertices is adjacent to p. This implies that $\mathcal{M}$ satisfies property (ii). Property (iii) follows from the fact that $|U \cup \{p\}| = |I_H| + 1$, so $\mathcal{Z}$ contains at most $2^{|I_H|+1}$ non-empty sets Z_X, each of which contributes at most one vertex to every set $M(x_i)$. Checking whether $\mathcal{M}$ also satisfies properties (iv) and (v) is done in exactly the same way as in Case 1. If $\mathcal{M}$ satisfies properties (iv) and (v), then $\mathcal{M}$ is an essential collection for U, and the algorithm concludes that U is H-compatible. If $\mathcal{M}$ does not satisfy both

properties (iv) and (v), then we unmark all the vertices in C_G and repeat the procedure on the next element of $\mathcal{B}$. If none of the elements of $\mathcal{B}$ yields an essential set for U, then the algorithm is repeated with another vertex $p' \in I_G \setminus U$ playing the role of p. If, for all the vertices of $I_G \setminus U$, none of the elements of $\mathcal{B}$ yields an essential collection for U, then the algorithm concludes that U is not H-compatible.

Let us argue why the algorithm for Case 2 is correct. If the algorithm finds a collection $\mathcal{M}$ that satisfies properties (i)–(v) of Lemma 3, then clearly $\mathcal{M}$ is an essential collection for U. Hence, by Lemma 3, our algorithm correctly concludes that U is H-compatible in this case. We now show that if U is H-compatible, then our algorithm will find an essential collection for U. Suppose U is H-compatible. By Lemma 3, there exists an essential collection $\mathcal{M}'$ for U. If there exists an essential collection $\mathcal{M}''$ for U such that every $M''(x_i) \in \mathcal{M}''$ contains at least one vertex of C_G, then the algorithm for Case 1 will find an essential collection for U by the arguments used in the correctness proof of Case 1. Suppose such a collection does not exist. Then, by property (i) of Lemma 3, we know that $\mathcal{M}'$ contains exactly one set $M'(x_j)$ such that $M'(x_j) = \{p\}$ for some $p \in I_G \setminus U$. All other sets of $\mathcal{M}'$ contain only vertices of C_G. Since p is not adjacent to any vertex of U, x_j must be a vertex of C_H that does not have any neighbors in I_H, i.e., $x_j \in S$. Recall that $x_k \in S$, and note that all the vertices of S have the same closed neighborhood in H. Hence we can assume that $x_j = x_k$, since we can swap the indices of x_j and x_k otherwise. Recall that the collection $\mathcal{Z}$ consists of all subsets of $U \cup \{p\}$ in Case 2. For $i = 1, \ldots, k-1$, the set $M'(x_i)$ contains at most one vertex from every set of $\mathcal{Z}$. For every vertex $x_i \in C_H \setminus \{x_k\}$, let A_i be the collection of sets in $\mathcal{Z}$ such that, for every $Z \in \mathcal{Z}$, $Z \in A_i$ if and only if $M'(x_i) \cap Z \neq \emptyset$. Let $B = (A_1, \ldots, A_{k-1})$. If our algorithm processes B, then it will find a collection of essential sets $M(x_1), \ldots, M(x_{k-1})$ such that, for $i = 1, \ldots, k-1$, the set $M(x_i)$ contains a vertex from exactly those sets in $\mathcal{Z}$ that $M'(x_i)$ contains a vertex from. Since $N_G(a) \cap (U \cup \{p\}) = N_G(b) \cap (U \cup \{p\})$ for every two vertices a and b in the same set $Z_X \in \mathcal{Z}$, $\mathcal{M} = \{M(x_1), \ldots, M(x_{k-1}), \{p\}\}$ satisfies properties (i)–(v) of Lemma 3. Hence $\mathcal{M}$ is an essential collection for U, and U is H-compatible.

It remains to determine the running time of our algorithm. In Case 1, the set $\mathcal{Z}$ contains all possible subsets of U, so $|\mathcal{Z}| = 2^{|U|} = 2^{|I_H|}$. Since $\mathcal{A}$ consists of all possible subsets of $\mathcal{Z}$, we have $|\mathcal{A}| = 2^{\mathcal{Z}} = 2^{2^{|I_H|}}$. Since $\mathcal{B}$ consists of all possible ordered $|C_H|$-tuples of elements of $\mathcal{A}$, we have $|\mathcal{B}| = |\mathcal{A}|^{|C_H|} = 2^{|C_H| \cdot 2^{|I_H|+1}}$. In the worst case, our algorithm tries all possible

elements of $\mathcal{B}$. Each of those elements yields a collection $\mathcal{M}$ of candidate essential sets. Testing whether $\mathcal{M}$ satisfies property (iv) can be done in $O(|V(G)|^2)$ time. To test whether $\mathcal{M}$ satisfies property (v), we first delete some vertices and contract each of the candidate essential sets into a single vertex, and then test whether the obtained graph is isomorphic to H. This can be done $O(|V(G)|^2)$ time and $2^{O(\sqrt{|V(H)|\log|V(H)|})}$ time [1], respectively. Hence the algorithm for Case 1 runs in $f(|V(H)|)\cdot|V(G)|^2$ time. In Case 2, the set $\mathcal{Z}$ contains all possible subsets of $U\cup\{p\}$, so $|\mathcal{Z}|=2^{|I_H|+1}$. Hence $|\mathcal{A}|=2^{\mathcal{Z}}=2^{2^{|I_H|+1}}$. Since $\mathcal{B}$ consists of all possible ordered $(|C_H|-1)$-tuples of elements of $\mathcal{A}$, we have $|\mathcal{B}|=|\mathcal{A}|^{|C_H|-1}=2^{(|C_H|-1)\cdot 2^{|I_H|+1}}$. The only other difference with Case 1 is that we might have to run the algorithm for Case 2 for every vertex $p\in I_G\setminus U$, which adds an $O(|I_G|)$ factor to the running time. Hence the total running time needed to test if the set U is H-compatible is $f(|V(H)|)\cdot|V(G)|^3$. □

Theorem 8 *Given a split graph G and an arbitrary graph H, it can be decided in $f(|V(H)|)\cdot|V(G)|^{O(\alpha(H))}$ time whether G can be contracted to H, where $\alpha(H)$ denotes the size of a maximum independent set in H and f is some function that does not depend on the size of G.*

Proof: Suppose we are given a split graph G, with split partition (C_G, I_G), and a graph H. Observe that contracting any edge of a split graph yields another split graph. Hence G can be contracted to H only if H is a split graph with $|V(G)|\geq|V(H)|$. We can check this in time linear in the size of H. Suppose H is a split graph, and let (C_H, I_H) be a split partition of H. By Lemma 2, G can be contracted to H if and only if G contains an H-compatible set. The number of different subsets of I_G of cardinality $|I_H|$ is $\binom{|I_G|}{|I_H|}\leq|V(G)|^{|I_H|}\leq|V(G)|^{\alpha(|H|)}$, where $\alpha(H)$ denotes the size of a maximum independent set in H. For each of those sets, we can test in $f(|V(H)|)\cdot|V(G)|^3$ time whether it is H-compatible by Lemma 4. Hence the overall running time is $f(|V(H)|)\cdot|V(G)|^{O(\alpha(H))}$. □

Theorem 8 immediately implies the following result.

Corollary 9 *For every fixed graph H, the problem of deciding whether a given split graph G can be contracted to H can be solved in polynomial time.*

We would like to mention that simultaneously and independent of our work, Golovach et al. [13] obtained a polynomial-time algorithm for H-Contractibility on split graphs for any fixed graph H (Theorem 6 in [13]). After proving an analogue of Lemma 2 (Lemma 1 in [13]),

they show how to test in $|V(G)|^{O(|C_H|)}$ time, for each set $U \subseteq I_G$ of size $|I_H|$, whether U is H-compatible. Thus they obtain an algorithm for Contractibility on split graphs that runs in time $|V(G)|^{O(|V(H)|)}$, yielding a polynomial-time algorithm for H-Contractibility for every fixed graph H. The main difference between our approach and the approach in [13] is that we use essential sets, which allows us to perform a more careful structural analysis of split graphs that can be contracted to a fixed split graph H. As a result, our algorithm can check whether a set U is H-compatible in $f(|V(H)|) \cdot |V(G)|^3$ time, whereas the algorithm in [13] needs $|V(G)|^{O(|C_H|)}$ time to perform this check. As stated in Theorem 8, we thus obtain an algorithm for Contractibility on split graphs that runs in time $f(|V(H)|) \cdot |V(G)|^{O(\alpha(H))}$, where $\alpha(H)$ denotes the size of a maximum independent set in H. In terms of parameterized complexity [8], Theorem 8 states that Contractibility is fixed-parameter tractable, with respect to parameter $|V(H)|$, when G is a split graph and H belongs to any graph class with bounded independence number. This complements the already known result that the problem is $W[1]$-hard (and therefore most likely not fixed-parameter tractable) with respect to this parameter when G is a split graph and H is a split graph with an arbitrarily large independence number [13].

5 Concluding Remarks

It is known that Induced Subgraph Isomorphism is NP-complete on cographs and on interval graphs [6, 9]. Hence Corollary 2 strengthens these existing NP-completeness results. The Induced Subgraph Isomorphism problem is also known to be NP-complete on another subclass of interval graphs, called proper interval graphs: given two proper interval graphs G and H, it is NP-complete to decide whether G has an induced subgraph that is isomorphic to H [6, 9]. However, the problem can be solved in polynomial time if the graph H is connected [16]. Thus we find it interesting that Induced Subgraph Isomorphism is NP-complete on *connected* trivially perfect graphs.

We presented an algorithm that solves H-Contractibility on split graphs in $f(|V(H)|) \cdot |V(G)|^{O(\alpha(H))}$ time for any split graph H, where $\alpha(H)$ denotes the size of a maximum independent set in H. As we mentioned in the introduction, it is unlikely that the problem can be solved in $f(|V(H)|) \cdot |V(G)|^{O(1)}$ time, since Golovach et al. [13] proved Contractibility to be $W[1]$-hard on split graphs when parameterized by $|V(H)|$. Is Contractibility fixed-parameter tractable on *interval* graphs when parameterized by $|V(H)|$, i.e., given two interval graphs G

and H, can we decide in time $f(|V(H)|) \cdot |V(G)|^{O(1)}$ whether G can be contracted to H?

References

[1] Babai, L., Luks, E. Canonical labelling of graphs. In: Proceedings of STOC 1983, pp. 171–183, ACM, New York (1983)

[2] Belmonte, R., Golovach, P. A., Heggernes, P., van 't Hof, P., Kamiński, M., Paulusma, D. Finding contractions and induced minors in chordal graphs via disjoint paths. In: Proceedings of ISAAC 2011, LNCS, vol. 7074, Springer, 2011, pp. 110–119.

[3] Belmonte, R., Heggernes, P., van 't Hof, P. Edge contractions in subclasses of chordal graphs. In: Proceedings of TAMC 2011, LNCS, vol. 6648, pp. 528–539, Springer (2011)

[4] Brandstädt, A., Le, V. B., Spinrad, J. Graph Classes: A Survey. SIAM Monographs on Discrete Mathematics and Applications (1999)

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

[6] Damaschke, P. Induced subgraph isomorphism for cographs is NP-complete. In: Proceedings of WG 1990, LNCS, vol. 484, pp. 72–78, Springer (1991)

[7] Diestel, R. Graph Theory (fourth edition). Springer-Verlag, Heidelberg (2010)

[8] Downey, R.G., Fellows, M.R. Parameterized Complexity. Monographs in Computer Science, Springer-Verlag, New York (1999)

[9] Garey, M. R., Johnson, D. S. Computers and Intractability. W.H. Freeman and Co., New York (1979)

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

[11] George, A., Liu, J. W. Computer Solution of Large Sparse Positive Definite. Prentice Hall Professional Technical Reference (1981)

[12] Golovach, P. A., Kamiński, M., Paulusma, D. Contracting a chordal graphs to a split graph or a tree. In: Proceedings of MFCS 2011, LNCS, vol. 6907, pp. 339–350 (2011)

[13] Golovach, P. A., Kamiński, M., Paulusma, D., Thilikos, D. M. Containment relations in split graphs. Discrete Applied Mathematics 160:155–163 (2012)

[14] Golumbic M.C. Algorithmic Graph Theory and Perfect Graphs. Annals of Discrete Mathematics 57, Elsevier (2004)

[15] Heggernes, P., Kratsch, D. Linear-time certifying recognition algorithms and forbidden induced subgraphs. Nordic Journal of Computing 14:87–108 (2007)

[16] Heggernes, P., Meister, D., Villanger, Y. Induced Subgraph Isomorphism on interval and proper interval graphs. In: Proceedings of ISAAC 2010, LNCS, vol. 6507, pp. 399–409, Springer (2010)

[17] van 't Hof, P., Kamiński, M., Paulusma, P., Szeider, S., Thilikos, D. M. On contracting graphs to fixed pattern graphs. In: Proceedings of SOFSEM 2010, LNCS, vol. 5901, pp. 503–514, Springer (2010)

[18] Kamiński, M., Paulusma, D., Thilikos, D. M. Contractions of planar graphs in polynomial time. In: Proceedings of ESA 2010, LNCS, vol. 6346, pp. 122–133, Springer (2010)

[19] Levin, A., Paulusma, D., Woeginger, G. J. The computational complexity of graph contractions I: polynomially solvable and NP-complete cases. Networks 51:178–189 (2008)

[20] Levin, A., Paulusma, D., Woeginger, G. J. The computational complexity of graph contractions II: two tough polynomially solvable cases. Networks 52:32–56 (2008)

[21] Lovász, L. Coverings and colorings of hypergraphs. In: Proceedings of the 4th Southeastern Conference on Combinatorics, Graphs Theory, and Computing, pp. 3–12, Utilitas Mathematica Publishing, Winnipeg (1973)

[22] Mahadev, N., Peled, U. Threshold graphs and related topics. Annals of Discrete Mathematics 56, North Holland (1995)

[23] Matoušek, J., Thomas, R. On the complexity of finding iso- and other morphisms for partial k-trees. Discrete Mathematics 108(1–3):343–364 (1992)

[24] Robertson, N., Seymour P.D. Graph Minors XIII. The Disjoint Paths Problem. Journal of Combinatorial Theory, Series B 63(1) (1995)

[25] Semple, C., Steel M. Phylogenetics. Oxford University Press graduate series Mathematics and its Applications (2003)

[26] Wolk, E. S. The comparability graph of a tree. In: Proceedings of the American Mathematical Society, vol. 13, pp. 789–795 (1962)

[27] Wolk, E. S. A note on "The comparability graph of a tree". Proceedings of the American Mathematical Society, vol. 16, pp. 17–20 (1965)

[28] Yan, J.-H., Chen, J.-J., Chang, G. J. Quasi-threshold graphs. Discrete Applied Mathematics 69:247–255 (1996)

Chapter 7

Detecting Fixed Patterns in Chordal Graphs in Polynomial Time

Detecting Fixed Patterns in Chordal Graphs in Polynomial Time

Rémy Belmonte[1] Petr A. Golovach[2] Pinar Heggernes[1]
Pim van 't Hof[1] Marcin Kamiński[3] Daniël Paulusma[2]

Department of Informatics, University of Bergen, Norway
{remy.belmonte,pinar.heggernes,pim.vanthof}@ii.uib.no

Département d'Informatique, Université Libre de Bruxelles, Belgium
Institute of Computer Science, University of Warsaw, Poland
mjk@mimuw.edu.pl

School of Engineering and Computing Sciences, Durham University, UK
{petr.golovach,daniel.paulusma}@durham.ac.uk

Abstract

The Contractibility problem takes as input two graphs G and H, and the task is to decide whether H can be obtained from G by a sequence of edge contractions. The Induced Minor and Induced Topological Minor problems are similar, but the first allows both edge contractions and vertex deletions, whereas the latter allows only vertex deletions and vertex dissolutions. All three problems are NP-complete, even for certain *fixed* graphs H. We show that these problems can be solved in polynomial time for every fixed H when the input graph G is chordal. Our results can be considered tight, since these problems are known to be W[1]-hard on chordal graphs when parameterized by the size of H. To solve Contractibility and Induced Minor, we define and use a generalization of the classic Disjoint Paths problem, where we require the vertices of each of the k paths to be chosen from a specified set. We prove that this variant is NP-complete even when $k = 2$, but that it is polynomial-time solvable on chordal graphs for every fixed k. Our algorithm for Induced Topological Minor is based on another generalization of Disjoint Paths called Induced Disjoint Paths, where the vertices from different paths may no longer be adjacent. We show that this problem, which is known to be NP-complete when $k = 2$, can be solved in polynomial time on chordal graphs even when k is part of the input. Our results fit into

the general framework of graph containment problems, where the aim is to decide whether a graph can be modified into another graph by a sequence of specified graph operations. Allowing combinations of the four well-known operations edge deletion, edge contraction, vertex deletion, and vertex dissolution results in the following ten containment relations: (induced) minor, (induced) topological minor, (induced) subgraph, (induced) spanning subgraph, dissolution, and contraction. Our results, combined with existing results, settle the complexity of each of the ten corresponding containment problems on chordal graphs.

1 Introduction

We study algorithmic problems that aim to decide whether the structure of a graph H appears as a "pattern" within the structure of another graph G. The exact definition of "pattern" depends on the graph operations that are allowed when modifying G into H. We consider the following four elementary graph operations. The operations *vertex deletion* (VD) and *edge deletion* (ED) simply remove a vertex or an edge, respectively, from the graph. The *edge contraction* (EC) operation, when applied to an edge uv, deletes the vertices u and v from the graph, and replaces them by a new vertex that is made adjacent to precisely those vertices to which u or v were adjacent. The *vertex dissolution* (VDi) operation can be applied to a vertex v of degree 2 whose two neighbors are not adjacent; it contracts one of the two edges incident with v. Table 1 shows ten graph containment relations obtained by combining these four operations. For example, a graph H is an induced minor of a graph G if H can be obtained from G by a sequence of vertex deletions and edge contractions (and consequently also vertex dissolutions), but not edge deletions. The corresponding decision problem, in which G and H form the ordered input pair (G, H), is called Induced Minor. The other rows in Table 1 are to be interpreted similarly.

With the exception of Graph Isomorphism, all problems in Table 1 are known to be NP-complete (cf. [29]). By the results of Robertson and Seymour [33], Grohe et al. [21], and Golovach et al. [19], Minor, Topological Minor, and Dissolution are in FPT with parameter $|V_H|$. The problems Spanning Subgraph Isomorphism and Graph Isomorphism require the input graphs G and H to have the same size, and hence they are trivially in FPT with parameter $|V_H|$. The problems Subgraph Isomorphism and Induced Subgraph Isomorphism are trivially in XP, as they can be solved by brute force checking all vertex subsets of G of size $|V_H|$. However, both problems are W[1]-hard with

Containment Relation	VD	ED	EC	VDi	Decision Problem
Minor	yes	yes	yes	yes	Minor
Induced minor	yes	no	yes	yes	Induced Minor
Topological minor	yes	yes	no	yes	Topological Minor
Induced topological minor	yes	no	no	yes	Induced Topological Minor
Contraction	no	no	yes	yes	Contractibility
Dissolution	no	no	no	yes	Dissolution
Subgraph	yes	yes	no	no	Subgraph Isomorphism
Induced subgraph	yes	no	no	no	Induced Subgraph Isomorphism
Spanning subgraph	no	yes	no	no	Spanning Subgraph Isomorphism
Isomorphism	no	no	no	no	Graph Isomorphism

Table 1: Ten known containment relations in terms of the four mentioned graph operations. The missing two combinations "no yes yes yes", and "no yes no yes" correspond to the minor and topological minor relations, respectively, if we allow an extra operation that removes isolated vertices.

parameter $|V_H|$, as they both generalize the well-known Clique problem, which is known to be W[1]-complete when parameterized by the size of the desired clique [10]. In summary, all of the above problems are polynomial-time solvable for every fixed graph H.

In contrast, the problems that we focus on in this paper are harder. In particular, there exist graphs H such that the three problems H-Contractibility, H-Induced Minor, and H-Induced Topological Minor are NP-complete, where the "H-" in front of the problem names indicates the variant of the problems where the graph H is fixed and only G is part of the input. Moreover, the problems Contractibility, Induced Minor, and Induced Topological Minor are W[1]-hard with parameter $|V_H|$ even on chordal graphs. We prove that these problems are in XP with parameter $|V_H|$ on chordal graphs. This implies that they can be solved in polynomial time on chordal graphs for every fixed graph H, as the class of chordal graphs is closed under edge contractions. Before we explain our results in more detail in the next section, we give an overview of known results on these problems.

For H-Contractibility, polynomial-time solvable and NP-complete cases, depending on H, can be found in a series of papers started by Brouwer and Veldman [7], followed by Levin et al. [27, 28], and Van 't Hof et al. [22]. The smallest NP-complete cases are when H is a path or a cycle on 4 vertices [7]. Fellows et al. [13] gave polynomial-time solvable and NP-complete cases for H-Induced Minor. The smallest known NP-complete case is a graph H on 68 vertices [13]. Even the question whether this problem is polynomial-time solvable for every fixed tree H is still open. Lévêque et al. [26] gave both polynomial-time solvable and NP-complete cases for H-Induced Topological Minor. This problem is NP-complete when H is a complete graph on 5 vertices, but its complexity is open when H is a complete graph on 4 vertices.

The following results, where $|V_H|$ is the parameter, are known for the case where the input graph G has a particular structure. The problems Contractibility and Induced Minor are in FPT on planar graphs by the respective results of Kamiński and Thilikos [24], and Fellows et al. [13]. Fiala et al. [11] showed that Induced Topological Minor is in XP on claw-free graphs, whereas Contractibility remains NP-complete on claw-free graphs even when H is a path on seven vertices [12]. Belmonte et al. [2] and Golovach et al. [19] independently proved that Contractibility is in XP on split graphs, which form a proper subclass of chordal graphs. In fact, all ten problems of Table 1 can be solved in polynomial time on split graphs for every fixed H [19]. However, whereas six of these problems are in FPT on split graphs, our three problems

Contractibility, Induced Minor, Induced Topological Minor, as well as the problem Induced Subgraph Isomorphism, are W[1]-hard on split graphs, and hence on chordal graphs [19]. This motivates our study of these three problems on chordal graphs with respect to XP algorithms.

Chordal graphs are the graphs in which every cycle of length at least four contains a chord. Chordal graphs constitute one of the most famous and well-studied graph classes, as they have a large number of practical applications in fields like sparse matrix computations, computational biology, computer vision, and VLSI design (cf. [17, 20, 34]). This graph class properly contains other well-known graph classes, like forests, interval graphs, and split graphs (cf. [6, 32]).

2 Our Results and Methodology

Table 2 gives a survey on the classical and the parameterized complexity of the ten containment problems from Table 1 on chordal graphs. We replaced "chordal" by "general" or "split" wherever possible in order to present the results in their strongest form. All the results on chordal graphs in Table 2 are new, and we prove them in the remainder of this paper. The remaining results in the table, namely the ones on split graphs and general graphs, are known and have already been mentioned. Section 3 contains the necessary additional terminology and a (straightforward) proof showing that Minor, Topological Minor, and Subgraph Isomorphism are in linear-time FPT with parameter $|V_H|$ on chordal graphs.

The three results from Table 2 that are left to prove are that Contractibility, Induced Minor, and Induced Topological Minor are in XP with parameter $|V_H|$ on chordal graphs. In order to obtain these results, we design algorithms for solving two generalizations of the classical Disjoint Paths problem on chordal graphs; these might be of interest independently of the studied graph containment problems.

In order to solve Contractibility and Induced Minor, we first need to solve some other problems. In Section 4, we introduce the following generalization of the Disjoint Paths problem. A *terminal pair* in a graph $G = (V, E)$ is a specified pair of vertices s and t called *terminals*, and the *domain* of a terminal pair (s, t) is a specified subset $U \subseteq V$ containing both s and t. We say that two paths, each of which is between some terminal pair, are *vertex-disjoint* if they have no common vertices except possibly the vertices of the terminal pairs. This leads to the following decision problem.

Containment Problem	Parameter: $\lvert V_H \rvert$
Minor	linear-time FPT on chordal; cubic-time FPT in general
Induced Minor	XP on chordal; W[1]-hard on split
Topological Minor	linear-time FPT on chordal; cubic-time FPT in general
Induced Topological Minor	XP on chordal; W[1]-hard on split
Contractibility	XP on chordal; W[1]-hard on split
Dissolution	linear-time FPT in general
Subgraph Isomorphism	linear-time FPT on chordal; cubic-time FPT in general
Induced Subgraph Isomorphism	XP in general; W[1]-hard on split
Spanning Subgraph Isomorphism	constant-time FPT in general
Graph Isomorphism	constant-time FPT in general

Table 2: The parameterized complexity of the ten problems from Table 1 on general graphs, chordal graphs and split graphs. All the results on chordal graphs are new.

Set-Restricted Disjoint Paths
Instance: A graph G, k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in G, and their respective domains $U_1, \ldots, U_k$.
Question: Does G contain k pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i is a path between s_i and t_i using only vertices from U_i, for $i = 1, \ldots, k$?

Note that the domains $U_1, \ldots, U_k$ are not necessarily pairwise disjoint. If we let every domain contain all vertices of G, we obtain exactly the Disjoint Paths problem.

The Set-Restricted Disjoint Paths problem is NP-complete on chordal, even interval, graphs, since Disjoint Paths is NP-complete on interval graphs [30]. Robertson and Seymour [33] showed that Disjoint Paths is in FPT with parameter k; their algorithm runs in $O(|V_G|^3)$ time for every fixed k. In contrast, we show that the more general Set-Restricted Disjoint Paths problem is NP-complete even when $k = 2$. We prove that on chordal graphs Set-Restricted Disjoint Paths is in XP with parameter k. We then consider disjoint trees, or equivalently, disjoint connected subgraphs, instead of disjoint paths. This leads to the following problem.

Set-Restricted Disjoint Connected Subgraphs
Instance: A graph G, k pairwise disjoint nonempty vertex subsets $S_1, \ldots, S_k$ of G, and their respective domains $U_1, \ldots, U_k$.
Question: Does G contain k pairwise vertex-disjoint connected subgraphs $G_1, \ldots, G_k$ such that $S_i \subseteq V_{G_i} \subseteq U_i$, for $1 \leq i \leq k$?

Choosing each domains U_i to be V_G yields the Disjoint Connected Subgraphs problem, which was introduced by Robertson and Seymour in [33]. This problem was shown to be NP-complete even when $k = 2$ and $\min\{|S_1|, |S_2|\} = 2$ [23]. Moreover, it is NP-complete on split graphs, and hence on chordal graphs, when $k = 2$. Robertson and Seymour [33] showed that it is in FPT with parameter $|S_1| + |S_2| + \cdots + |S_k|$. We show that the more general Set-Restricted Disjoint Connected Subgraphs problem is in XP with this parameter when restricted to chordal graphs.

In Section 5, we show how to use our XP algorithm for the problem Set-Restricted Disjoint Connected Subgraphs as a subroutine for solving Contractibility and Induced Minor on chordal graphs in time $|V_G|^{O(|V_H|^2)}$.

In Section 6, we turn our attention to Induced Topological Minor. To solve this problem, we use another generalization of Disjoint

Paths. We say that two paths P_1 and P_2 between two terminal pairs in a graph $G = (V, E)$ are *mutually induced* if they are vertex-disjoint and no vertex of P_1 is adjacent to a vertex of P_2, except possibly the terminal vertices. Note that each path is not necessarily induced (chordless), but we may assume this without loss of generality. This leads to the following decision problem.

Induced Disjoint Paths
Instance: A graph G and k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in G.
Question: Does G contain k mutually induced paths $P_1, \ldots, P_k$ such that P_i is a path between s_i and t_i, for $i = 1, \ldots, k$?

The Induced Disjoint Paths problem is already NP-complete for $k = 2$, due to a result of Bienstock [3]. Hence, on general graphs this problem is harder than Disjoint Paths, which is in FPT with parameter k as mentioned earlier. Note that on general graphs, Induced Disjoint Paths generalizes Disjoint Paths; subdividing the edges of an input graph of the latter problem yields an instance of the former problem. However, this is not true on chordal graphs, as subdividing the edges of a chordal graph might result in a graph that is not chordal. Interestingly, on chordal graphs the two problems completely swap complexity: although Disjoint Paths is NP-complete on chordal graphs as mentioned above, we show that Induced Disjoint Paths is polynomial-time solvable on chordal graphs. We use the corresponding algorithm as a subroutine in our algorithm for solving Induced Topological Minor on chordal graphs in time $O(|E_H| \cdot |V_G|^{|V_H|+3})$.

In Section 7, we give another application of our algorithm for solving Set-Restricted Disjoint Paths by using it as a subroutine to solve the Set-Restricted Disjoint Connected Subgraphs problem on interval graphs. We conclude the paper by mentioning some open problems.

3 Preliminaries

All graphs considered in this paper are finite, undirected, and have neither self-loops nor multiple edges. All the problems in this paper have a graph G as a part of the input. Throughout the paper, we use n and m to denote the number of vertices and edges of this input graph G, respectively.

Let $G = (V, E)$ be a graph. If the vertex and edge sets of a graph G are not specified, we use V_G and E_G to denote these sets, respectively. A subset $U \subseteq V$ is a *clique* if every pair of vertices in U are adjacent. A

vertex is *simplicial* if its neighbors form a clique. We write $G[U]$ to denote the subgraph of G induced by $U \subseteq V$. Two sets $U, U' \subseteq V$ are called *adjacent* if there exist vertices $u \in U$ and $u' \in U'$ such that $uu' \in E$. A path between vertices u and v is called a (u, v)-path. The set of vertices of a path P is denoted by V_P.

The *subdivision* of an edge $e = uw$ in a graph removes e, adds a new vertex v and two new edges uv and vw. We say that a graph G is a *subdivision* of a graph H if G can be obtained from H by a sequence of edge subdivisions. We observe that an edge subdivision is the reverse operation of a vertex dissolution. Hence, H is an (induced) topological minor of G if and only if an (induced) subgraph of G is a subdivision of H.

An *H-witness structure* of G is a partition of V into $|V_H|$ nonempty sets $W(x)$, one set for each $x \in V_H$, called *H-witness sets*, such that

(i) each $W(x)$ induces a connected subgraph of G; and

(ii) for all $x, y \in V_H$ with $x \neq y$, sets $W(x)$ and $W(y)$ are adjacent in G if and only if x and y are adjacent in H.

Observe that H is a contraction of G if and only if G has an H-witness structure: H can be obtained from G by contracting the edges in each H-witness set until a single vertex remains in each of them. This view provides an intuition on the hardness of the Contractibility problem: it is a partition problem rather than a subset problem.

A *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{X}$ is a collection of subsets of V, called *bags*, and $\mathcal{T}$ is a tree whose vertices, called *nodes*, are the sets of $\mathcal{X}$, such that the following three properties are satisfied.

- $\bigcup_{X \in \mathcal{X}} X = V$,
- for each edge $uv \in E$, there is a bag $X \in \mathcal{X}$ with $u, v \in X$,
- for each $x \in V$, the set of nodes containing x forms a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{T}, \mathcal{X})$ is the size of a largest bag in $\mathcal{X}$ minus 1. The *treewidth* of G is the minimum width over all possible tree decompositions of G.

A *chord* of a path or a cycle is an edge between two non-consecutive vertices of the path or the cycle. A graph is *chordal* if every cycle of it on at least four vertices has a chord. It is not difficult to see that the class of chordal graphs is closed under vertex deletions and edge contractions but not under edge deletions. Chordal graphs can be recognized in linear

time [35]. Every chordal graph contains at most n maximal cliques, and, if it is not a complete graph, at least two non-adjacent simplicial vertices [9]. A graph G is chordal if and only if it has a tree decomposition whose set of bags is exactly the set of maximal cliques of G [16]. Such a tree decomposition is called a *clique tree* and can be constructed in linear time [4].

The complexity classes XP and FPT are defined in the framework of parameterized complexity. A parameterized problem Q belongs to the class XP if for each instance (I, k) it can be decided in $|I|^{f(k)}$ time whether $(I, k) \in Q$, where f is a function that depends only on the *parameter* k, and $|I|$ denotes the size of I. If a problem belongs to XP, then it can be solved in polynomial time for every fixed k. Hence, if a problem is NP-complete for some fixed value of k, then it is unlikely to belong to XP. If a parameterized problem can be solved by an algorithm with running time $f(k)\,|I|^{O(1)}$, then the problem belongs to the class FPT. A problem is in cubic-time FPT, linear-time FPT, or constant-time FPT if it can be solved in time $f(k)\,|I|^3$, in time $f(k)\,|I|$ or in time $f(k)$, respectively. Between FPT and XP is a hierarchy of parameterized complexity classes, $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{XP}$, where hardness for one of the W-classes is considered to be strong evidence of intractability with respect to the class FPT. For formal background on parameterized complexity, we refer to the textbooks by Downey and Fellows [10], Flum and Grohe [14], and Niedermeier [31].

Let u, v, w be three distinct vertices in a graph such that uv and vw are edges. The operation that removes the edges uv and vw, and adds the edge uv in the case u and w are not adjacent, is called a *lift*. A graph G contains H as a *immersion* if H can be obtained from G by a sequence of vertex deletions, edge deletions, and lifts. The corresponding decision problem is called Immersion. Grohe et al. [21] showed that Immersion is in FPT with parameter $|V_H|$; their running time is $O(n^3)$ for every fixed H. Proposition 1 below proves that this problem can be solved in linear time on chordal graphs, and also proves the linear-time FPT results on Minor, Topological Minor, and Subgraph Isomorphism that were stated in Table 2.

Proposition 1. *The problems* Minor, Topological Minor, Subgraph Isomorphism, *and* Immersion *can be solved in* $f(|V_H|) \cdot (n+m)$ *time on chordal graphs for some function* f *that depends only on* $|V_H|$.

Proof. Let G be a chordal input graph. A clique C of maximum size in G can be found in $O(n+m)$ time [35]. If $|C| \geq |V_H|$, then $G[C]$, and hence G, contains H as a subgraph and thus as a minor, topological minor, and

immersion. If $|C| < |V_H|$, then the treewidth of G is $|C| - 1 < |V_H| - 1$ by the definition of a clique tree. Because H is fixed, the treewidth of G is bounded by a constant. A seminal result of Courcelle [8] states that on every class of graphs of bounded treewidth, every problem expressible in monadic second-order logic, i.e., the fragment of second-order logic where quantified relation symbols must have arity 1, can be solved in $O(n)$ time. For fixed H, it is known that the problems H-Subgraph Isomorphism, H-Minor, H-Topological Minor, and H-Immersion can all be expressed in monadic second-order logic; in particular we refer to Grohe et al. [21] for the case of immersions. Since a clique tree can be constructed in $O(n+m)$ time, this completes the proof of Proposition 1. □

4 Set-Restricted Disjoint Paths

We start with the following result.

Theorem 2. Set-Restricted Disjoint Paths *is* NP*-complete even when* $k = 2$.

Proof. We reduce from the NP-complete 3-Sat problem [15]. It is well known that this problem remains NP-complete when each Boolean variable occurs at most twice as a positive literal and at most twice as a negative literal. We use this variant for our reduction. Let Φ be an instance of 3-Sat with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. We construct a graph G as follows; see also Figure 1.

- Add two vertices s and t.
- Add $n+1$ vertices $v_0, \ldots, v_n$ and edges sv_0 and v_nt.
- For $i = 1, \ldots, n$, add vertices $x_i^{(1)}, x_i^{(2)}, \overline{x}_i^{(1)}, \overline{x}_i^{(2)}, y_i, \overline{y}_i$ and edges $v_{i-1}x_i^{(1)}$, $x_i^{(1)}y_i$, $y_i x_i^{(2)}$, $x_i^{(2)}v_i$, $v_{i-1}\overline{x}_i^{(1)}$, $\overline{x}_i^{(1)}\overline{y}_i$, $\overline{y}_i\overline{x}_i^{(2)}$, $\overline{x}_i^{(2)}v_i$.
  Let $Q_i = v_{i-1}x_i^{(1)}y_i x_i^{(2)}v_i$ and $\overline{Q}_i = v_{i-1}\overline{x}_i^{(1)}\overline{y}_i\overline{x}_i^{(2)}v_i$.
- Add $m+1$ vertices $u_0, \ldots, u_m$ and edges su_0 and u_mt.
- For each clause C_j and each literal z in C_j:
  - if $z = x_i$, then add edges $u_{j-1}x_i^{(1)}$, $u_j x_i^{(1)}$ if z is the first occurrence of the literal x_i in Φ, and edges $u_{j-1}x_i^{(2)}$, $u_j x_i^{(2)}$ if z is the second occurrence.

– if $z = \overline{x}_i$, then add edges $u_{j-1}\overline{x}_i^{(1)}$, $u_j\overline{x}_i^{(1)}$ if z is the first occurrence of the literal $\overline{x}_i$ in Φ, and edges $u_{j-1}\overline{x}_i^{(2)}$, $u_j\overline{x}_i^{(2)}$ if z is the second occurrence.

Let $R_j(z)$ be the obtained (u_{j-1}, u_j)-path of length two.

- Let $U_1 = V_G \setminus \{u_0, \ldots, u_m\}$.
- Let $U_2 = \{s, t\} \cup \{u_0, \ldots, u_m\} \cup \{x_1^{(1)}, \ldots, x_n^{(1)}\} \cup \{x_1^{(2)}, \ldots, x_n^{(2)}\}$.

We prove that Φ can be satisfied if and only if there are two vertex-disjoint (s, t)-paths P_1 and P_2 in G such that $V_{P_1} \subseteq U_1$ and $V_{P_2} \subseteq U_2$; recall that we allow such paths to have common end-vertices, as is the case here.

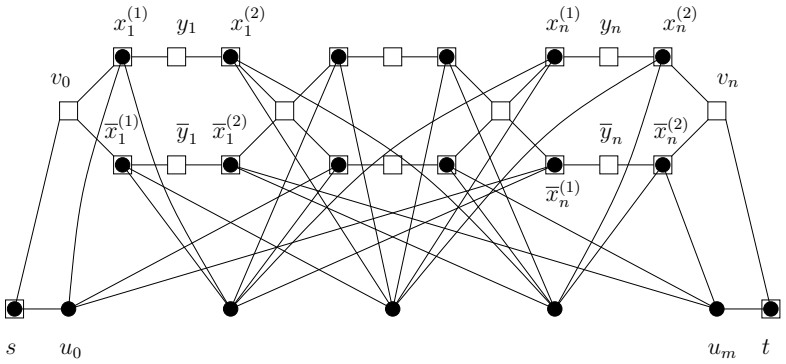


Figure 1: The graph G; the vertices of U_1 and U_2 are shown by white squares and black circles, respectively. A white square with a black circle inside indicates that the vertex belongs to both sets.

First suppose that the variables $x_1, \ldots, x_n$ have a truth assignment that satisfies Φ. We construct P_1 as follows. For each $i \in \{1, \ldots, n\}$, we choose Q_i if $x_i = \texttt{false}$, and $\overline{Q}_i$ if $x_i = \texttt{true}$. Afterwards, we concatenate the chosen paths. We get a (v_0, v_n)-path, and construct the (s, t)-path P_1 by adding the edges sv_0 and v_nt. By construction, $V_{P_1} \subseteq U_1$. We construct P_2 as follows. For each $j \in \{1, \ldots, m\}$, the clause C_j can be assumed to contain a literal $z = \texttt{true}$, and we select such a literal and the corresponding paths $R_j(z)$. Afterwards, we concatenate the chosen paths. We get a (u_0, u_m)-path, and construct the (s, t)-path P_2 by adding the edges su_0 and u_mt. By construction, $V_{P_2} \subseteq U_2$. If P_2 contains $R_j(z) = u_{j-1}x_i^{(1)}u_j$ or $R_j(z) = u_{j-1}x_i^{(2)}u_j$ as a subpath, then $x_i = \texttt{true}$ implying that $\overline{Q}_i$ is a subpath of P_1. Hence, $x_i^{(1)}, x_i^{(2)} \notin V_{P_1}$. We conclude that P_1 and P_2 are vertex-disjoint.

Now suppose that there are vertex-disjoint (s,t)-paths P_1 and P_2 in G, such that $V_{P_1} \subseteq U_1$ and $V_{P_2} \subseteq U_2$. Note that for each $i \in \{1,\ldots,n\}$, either Q_i or $\overline{Q}_i$ is a subpath of P_1. If Q_i is a subpath of P_1, then we set $x_i = \texttt{false}$, and $x_i = \texttt{true}$ otherwise. Note that for each $j \in \{1,\ldots,m\}$, the path P_2 contains $R_j(z)$ as a subpath for some literal z in C_j. If $z = x_i$ for some variable x_i, then our assumption that P_1 and P_2 are vertex-disjoint implies that $\overline{Q}_i$ is a subpath of P_1. Hence, $x_i = \texttt{true}$, and consequently $z = \texttt{true}$. Similarly, if $z = \overline{x}_i$ for some variable x_i, then Q_i is a subpath of P_1 and $x_i = \texttt{false}$, and consequently, $z = \texttt{true}$. Hence, each clause C_j is satisfied by this truth assignment, and $\Phi = \texttt{true}$, as desired. This completes the proof of Theorem 2. □

We apply dynamic programming to prove that Set-Restricted Disjoint Paths can be solved in polynomial time on chordal graphs for every fixed integer k. The first key observation is that the existence of k disjoint paths is equivalent to the existence of k disjoint *induced*, i.e. chordless, paths. The second key observation is that every induced path contains at 0, 1, or 2 vertices from each clique. Our algorithm solves the decision problem, but it can easily be modified to produce the desired paths if they exist.

Kloks [25] showed that every tree decomposition of a graph can be converted in linear time to a *nice tree decomposition*, such that the size of the largest bag does not increase, and the total size of the tree is linear in the size of the original tree. A tree decomposition $(\mathcal{T}, \mathcal{X})$ is *nice* if $\mathcal{T}$ is a binary tree with root X_r such that the nodes of $\mathcal{T}$ are of the following four types:

1. a *leaf node* X is a leaf of $\mathcal{T}$ and has size $|X| = 1$;
2. an *introduce node* X has one child X' with $X = X' \cup \{v\}$ for some vertex $v \in V_G$;
3. a *forget node* X has one child X' with $X = X' \setminus \{v\}$ for some vertex $v \in V_G$;
4. a *join node* X has two children X' and X'' with $X = X' = X''$.

Applying the conversion algorithm of Kloks [25] on a clique tree of a chordal graph G leads to a nice tree decomposition of G with the additional property that each bag is a (not necessary maximal) clique in G. We use such nice tree decompositions of chordal graphs in our algorithm for Set-Restricted Disjoint Paths, which we describe next.

Let k be a positive integer, and let G be a chordal graph with k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ that have domains $U_1, \ldots, U_k$, respectively. If G is disconnected, we check for each terminal pair (s_i, t_i) whether s_i and t_i belong to the same connected component. If not, then we return `No`. Otherwise, we consider each connected component and its set of terminals separately. Consequently, we assume from now on that G is connected.

We construct a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ of G with root X_r, such that each bag is a clique in G. For every node $X_i \in V_{\mathcal{T}}$, we denote by $\mathcal{T}_i$ the subtree of $\mathcal{T}$ with root X_i induced by X_i and all its descendants. We define $G_i = G[\bigcup_{j \in V_{\mathcal{T}_i}} X_j]$, i.e., the subgraph of G induced by the set of all vertices of G appearing in bags of $\mathcal{T}_i$.

Our dynamic programming algorithm keeps a table for each node of $\mathcal{T}$. For a node X_i, the table stores a collection of records

$$\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k)),$$

where $R_1, \ldots, R_k \subseteq X_i$ are ordered sets without common vertices except (possibly) terminals, where $R_j \subseteq U_j$ and $0 \leq |R_j| \leq 2$ for $j \in \{1, \ldots, k\}$, and where each $State_j$ can have one of the following four values:

Not initialized, *Started from* s, *Started from* t, or *Completed*.

These records correspond to the partial solution of Set-Restricted Disjoint Paths for G_i with the following properties.

- If $State_j =$ *Not initialized*, then $s_j, t_j \notin V_{G_i}$. If $R_j = \emptyset$, then (s_j, t_j)-paths have no vertices in G_i in the partial solution. If $R_j = \langle z \rangle$, then z is the unique vertex of a (s_j, t_j)-path in G_i in the partial solution. If $R_j = \langle z_1, z_2 \rangle$, then z_1, z_2 are vertices in a (s_j, t_j)-path, z_1 is the predecessor of z_2 in the path, and this path has no other vertices in G_i.

- If $State_j =$ *Started from* s, then $s_j \in V_{G_i}$, $t_j \notin V_{G_i}$ and R_j contains either one or two vertices. If $R_j = \langle z \rangle$, then the partial solution contains an (s_j, z)-path with the unique vertex $z \in X_i$. If $R_j = \langle z_1, z_2 \rangle$, then the partial solution contains an (s_j, z_2)-path such that z_1 is the predecessor of z_2 with exactly two vertices $z_1, z_2 \in X_i$.

- If $State_j =$ *Started from* t, then $s_j \notin V_{G_i}$, $t_j \in V_{G_i}$ and R_j contains either one or two vertices. If $R_j = \langle z \rangle$, then the partial solution contains a (z, t_j)-path with the unique vertex $z \in X_i$. If $R_j = \langle z_1, z_2 \rangle$, then the partial solution contains an (z_1, t_j)-path such that z_2 is the successor of z_1 with exactly two vertices $z_1, z_2 \in X_i$.

- If $State_j = \textit{Completed}$, then $s_j \in V_{G_i}$, $t_j \in V_{G_i}$. The partial solution in this case contains an (s_j, t_j)-path, and R_j is the set of vertices of this path in X_i. If $R_j = \langle z_1, z_2 \rangle$, then z_1 is the predecessor of z_2 in the path.

The tables are constructed and updated as follows.

Leaf nodes. Let $X_i = \{u\}$ be a leaf node of $\mathcal{T}$. Then the table for X_i stores all records $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ with the following properties. If u is a terminal, then for $j = 1, \ldots, k$,

- if $u = s_j$, then $State_j = \textit{Started from } s$ and $R_j = \langle u \rangle$;
- if $u = t_j$, then $State_j = \textit{Started from } t$ and $R_j = \langle u \rangle$; and
- if $u \neq s_j$ and $u \neq t_j$, then $State_j = \textit{Not initialized}$ and $R_j = \emptyset$.

If u is not a terminal, then $State_j = \textit{Not initialized}$ for $j = 1, \ldots, k$, and either $R_j = \emptyset$ for all $j \in \{1, \ldots, k\}$, or exactly one set $R_j = \langle u \rangle$ if $u \in U_j$ and other sets are empty.

Introduce nodes. Let X_i be an introduce node with child $X_{i'}$, and let $X_i = X_{i'} \cup \{u\}$ for some $u \in V_G$. We consider two cases.

Case 1. u is a terminal.
For each record $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$, we either modify $\mathcal{R}'$ and include the modified record $\mathcal{R} = ((State_1, R_1)$ $, \ldots, (State_k, R_k))$ in the table for X_i, or we discard $\mathcal{R}'$. If there exists an index j such that $u \in \{s_j, t_j\}$ and $|R'_j| = 2$, then $\mathcal{R}'$ is discarded. Otherwise,

- if $u = s_j$ and $State'_j = \textit{Not initialized}$, then $State_j = \textit{Started from } s$; if $R'_j = \emptyset$, then $R_j = \langle u \rangle$, and if $R'_j = \langle z \rangle$, then $R_j = \langle u, z \rangle$;
- if $u = s_j$ and $State'_j = \textit{Started from } t$, then $State_j = \textit{Completed}$; if $R'_j = \langle z \rangle$, then $R_j = \langle z, u \rangle$;
- if $u = t_j$ and $State'_j = \textit{Not initialized}$, then $State_j = \textit{Started from } t$; if $R'_j = \emptyset$, then $R_j = \langle u \rangle$, and if $R'_j = \langle z \rangle$, then $R_j = \langle z, u \rangle$;
- if $u = t_j$ and $State'_j = \textit{Started from } s$, then $State_j = \textit{Completed}$; if $R'_j = \langle z \rangle$ then $R_j = \langle z, u \rangle$; and
- if $u \neq s_j$ and $u \neq t_j$, then $State_j = State'_j$ and $R_j = R'_j$.

Case 2. u is not a terminal.
We include all records from the table for $X_{i'}$ in the table for X_i. In addition, we add new records to the table for X_i according to the following rules. For each $j \in \{1, \ldots, k\}$ with $u \in U_j$, we consider the records $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$ with $|R'_j| \leq 1$, and do the following:

- if $State'_j = \textit{Not initialized}$ and $R'_j = \langle z \rangle$, then we add to the table for X_i two records such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle z, u \rangle$ for the first record and $R_j = \langle u, z \rangle$ for the second;
- if $State'_j = \textit{Not initialized}$ and $R'_j = \emptyset$, then we include in the table for X_i the record such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle u \rangle$;
- if $State'_j = \textit{Started from } s$ and $R'_j = \langle z \rangle$, then we add to the table for X_i the record such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle z, u \rangle$;
- if $State'_j = \textit{Started from } t$ and $R'_j = \langle z \rangle$, then we add to the table for X_i the record such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle u, z \rangle$.

Forget nodes. Let X_i be a forget node with child $X_{i'}$, and let $X_i = X_{i'} \setminus \{u\}$ for some $u \in V_G$. Every record $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$ with $u \notin R'_j$ for all $j \in \{1, \ldots, k\}$ is included in the table for X_i. For each record $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$ such that $u \in R'_j$ for some $j \in \{1, \ldots, k\}$, we either modify it and include the modified record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ in the table for X_i, or we discard $\mathcal{R}'$, using the following rules:

- for all $j \in \{1, \ldots, k\}$, $State_j = State'_j$;
- for $j \in \{1, \ldots, k\}$, if $u \notin R'_j$, then $R_j = R'_j$;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j = \textit{Started from } s$, then we discard the record if $R'_j = \langle u \rangle$ or $R'_j = \langle z, u \rangle$, and we set $R_j = \langle z \rangle$ if $R'_j = \langle u, z \rangle$;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j = \textit{Started from } t$, then we discard the record if $R'_j = \langle u \rangle$ or $R'_j = \langle u, z \rangle$, and we set $R_j = \langle z \rangle$ if $R'_j = \langle z, u \rangle$;

- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j = \textit{Not initialized}$, then we discard the record;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j = \textit{Completed}$, then $R_j = R'_j \setminus \langle u \rangle$.

Join nodes. Let X_i be a join node with children $X_{i'}$ and $X_{i''}$. For each pair of records $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ and $\mathcal{R}'' = ((State''_1, R''_1), \ldots, (State''_k, R''_k))$ from the tables for $X_{i'}$ and $X_{i''}$, respectively, such that $R'_j = R''_j$ for all $j \in \{1, \ldots, k\}$, we construct the record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ and include it in the table for X_i:

- for $j = 1, \ldots, k$, $R_j = R'_j = R''_j$;
- for $j = 1, \ldots, k$, if $State'_j = State''_j$, then $State_j = State'_j = State''_j$;
- for $j = 1, \ldots, k$, if $State'_j = \textit{Not initialized}$, then $State_j = State''_j$;
- for $j = 1, \ldots, k$, if $State''_j = \textit{Not initialized}$, then $State_j = State'_j$;
- for $j = 1, \ldots, k$, if $State'_j = \textit{Completed}$ or $State''_j = \textit{Completed}$, then $State_j = \textit{Completed}$;
- for $j = 1, \ldots, k$, if $State'_j = \textit{Started from } s$ and $State''_j = \textit{Started from } t$ or $State'_j = \textit{Started from } t$ and $State''_j = \textit{Started from } s$, then $State_j = \textit{Completed}$.

The algorithm computes these tables for all nodes of $\mathcal{T}$, starting from the leaves. Finally, the table for the root X_r is constructed. The algorithm returns `Yes` if the table for X_r contains a record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ with $State_1 = \ldots = State_k = \textit{Completed}$, and it returns `No` otherwise.

Theorem 3. *The* Set-Restricted Disjoint Paths *problem can be solved in $n^{O(k)}$ time on chordal graphs.*

Proof. The correctness of the algorithm follows from its description, keeping in mind that we are looking for k disjoint *induced* paths, each of which contains at most two vertices of every clique. For the running time, recall that a clique tree can be constructed in linear time [4] and that it can be converted in linear time to a nice tree decomposition in which each bag corresponds to a clique [25]. It remains to observe that each table contains at most $n^{O(k)}$ records, since each R_j has at most two elements. Since there are $O(n)$ nodes in $\mathcal{T}$ and hence $O(n)$ tables in total, our algorithm runs in $n^{O(k)}$ time. □

For our purposes, we need to generalize Theorem 3 in the following way.

Corollary 1. *The problem* Set-Restricted Disjoint Connected Subgraphs *can be solved in* $n^{O(p)}$ *time on chordal graphs, where* $p = \sum_{i=1}^{k} |S_i|$ *is the sum of the sizes of the terminal sets.*

Proof. Let $G = (V, E)$ be a chordal graph on n vertices with terminal sets $S_1, \ldots, S_k$ and corresponding domains $U_1, \ldots, U_k$. To solve Set-Restricted Disjoint Connected Subgraphs on G, we check whether G contains pairwise vertex-disjoint trees $T_1, \ldots, T_k$, such that T_i contains all vertices of S_i and such that all its other vertices are from U_i for $i = 1, \ldots, k$. For this purpose, we simply generate all collections of k subsets of V that can possibly give us the desired subtrees $T_1, \ldots, T_k$. We need to argue that it is sufficient to generate collections that consist of $O(p)$ vertices.

Let $p_i = |S_i|$ for $i = 1, \ldots, k$. Observe that every inclusion minimal subtree T_i of G with $S_i \subseteq V_{T_i}$ contains at most $p_i - 2$ vertices of degree at least 3 that are not in S_i. Repeatedly contracting every edge in T_i that is incident with a vertex of degree 2 in T_i, results in a reduced tree T_i' with at most $2p_i - 2$ vertices, and consequently, at most $2p_i - 3$ edges. Every edge in T_i' corresponds to a path in G. Hence we can guess the $2p_i - 2$ vertices of each possible reduced tree T_i', and then expand each edge of the tree to a path in G if possible, to obtain every possible tree T_i, using our algorithm for Set-Restricted Disjoint Paths. Consequently, we proceed as follows.

We iterate over every collection of k pairwise disjoint sets $X_1, \ldots, X_k$ with $X_i \subseteq U_i \setminus S_i$ and $|X_i| \leq p_i - 2$ for $i = 1, \ldots, k$. In each collection, for each $S_i \cup X_i$, we generate the set $\mathcal{T}_i$ of all possible trees with vertex set $S_i \cup X_i$ such that the vertices of X_i are of degree at least 3. Every edge st in a tree from $\mathcal{T}_i$ gives us a terminal pair (s', t') with domain U_i. We now pick one tree T_i' for $i = 1, \ldots, k$. This leads to a choice of trees $T_1', \ldots, T_k'$, where each T_i' corresponds to a set of terminal pairs with domain U_i as we explained above. We call such a choice a *T'-combination*. Let $(s_1', t_1'), \ldots, (s_{k'}', t_{k'}')$ denote the union of these k sets of terminal pairs and denote the domain of (s_h', t_h') by U_h' for $h = 1, \ldots, k'$. We note that $U_h' = U_i$ if and only if (s_h', t_h') is a terminal pair corresponding to an edge in T_i'. We also note that $k' \leq \sum_{i=1}^{k}(2p_i - 3)$, because each T_i' has at most $2p_i - 3$ edges. We now solve the Set-Restricted Disjoint Paths problem for this instance. If we find a solution, then we return `Yes`. Otherwise, we choose another T'-combination and solve the resulting instance, until we have considered all T'-combinations.

In order to prove that the above procedure leads to a total running time of $n^{O(p)}$, we observe that there are $n^{O(p)}$ possibilities to choose the sets $X_1, \ldots, X_k$. Moreover, by Cayley's formula, we have at most $(2p_i - 2)^{2p_i-4} \leq p^{2p_i}$ different possibilities to join the vertices of $S_i \cup X_i$ by paths to obtain a tree. For each choice, we check the existence of at most $\sum_{i=1}^{k}(2p_i - 3) \leq 2p$ disjoint paths, which can be done in $n^{O(2p)}$ time by Theorem 3. Hence, the total running time is $n^{O(p)} \cdot p^{2p_1+\cdots+2p_k} \cdot n^{O(2p)} = n^{O(p)}$. This completes our proof. □

5 Contractions and Induced Minors in Chordal Graphs

First, in Section 5.1 below, we give a structural characterization of chordal graphs that contain a fixed graph H as a contraction. Then, in Section 5.2, we present our XP algorithm for solving Contractibility on chordal graphs, and show how it can be used to solve Induced Minor as well.

5.1 Properties

Throughout Section 5.1, let G be a connected chordal graph, let $\mathcal{T}_G$ be a clique tree of G, and let H be a graph with $V_H = \{x_1, \ldots, x_k\}$. For a set of vertices $A \subseteq V_G$, we let $G(A)$ denote the induced subgraph of G obtained by recursively deleting simplicial vertices that are not in A. Since every leaf in every clique tree contains at least one simplicial vertex, we immediately obtain Lemma 4 below. This lemma, in combination with Lemma 5, is crucial for the running time of our algorithm.

Lemma 4. *For every set $A \subseteq V_G$, every clique tree of $G(A)$ has at most $|A|$ leaves.*

Lemma 5. *The graph H is a contraction of G if and only if there is a set $A \subseteq V_G$ such that $|A| = k$ and H is a contraction of $G(A)$.*

Proof. First suppose that H is a contraction of G. Let $\mathcal{W}$ be an H-witness structure of G. For each $i \in \{1, \ldots, k\}$, we choose an arbitrary vertex $a_i \in W(x_i)$, and let $A = \{a_1, \ldots, a_k\}$. Suppose that G has a simplicial vertex $v \notin A$, and assume without loss of generality that $v \in W(x_1)$. Because $v \neq a_1$ and $a_1 \in W(x_1)$, we find that $|W(x_1)| \geq 2$. Hence, $W(x_1)$ contains a vertex u adjacent to v. The graph G', obtained from G by deleting v, is isomorphic to the graph obtained from G by contracting uv, since v is simplicial. Because u and v belong to the same witness set,

namely $W(x_1)$, this implies that H is a contraction of G'. Using these arguments inductively, we find that H is a contraction of $G(A)$.

Now suppose that A is a subset of V_G with $|A| = k$, and that H is a contraction of $G(A)$. Deleting a simplicial vertex v in a graph is equivalent to contracting an edge incident with v. This means that $G(A)$ is a contraction of G. Because H is a contraction of $G(A)$ and contractibility is a transitive relation, we conclude that H is a contraction of G as well. □

For a subtree $\mathcal{T}$ of $\mathcal{T}_G$, we say that a vertex $v \in V_G$ is an *inner* vertex for $\mathcal{T}$ if v only appears in the maximal cliques of G that are nodes of $\mathcal{T}$. By $I(\mathcal{T}) \subseteq V_G$ we denote the set of all inner vertices for $\mathcal{T}$. For a subset $S \subseteq V_G$, let $\mathcal{T}_S$ be the unique minimal subtree of $\mathcal{T}_G$ that contains all maximal cliques of G that have at least one vertex of S; we say that a vertex v is an *inner* vertex for S if $v \in I(\mathcal{T}_S)$, and we set $I(S) = I(\mathcal{T}_S)$. Lemma 7 below provides an alternative and useful structural description of G if it contains H as a contraction. We need the following lemma to prove Lemma 7.

Lemma 6. *Let $S \subseteq V_G$ and let T be a subgraph of G that is a tree such that $S \subseteq V_T \subseteq I(S)$. Then $K \cap V_T \neq \emptyset$ for each node K of $\mathcal{T}_S$.*

Proof. Let K be a node of $\mathcal{T}_S$. If $K \cap S \neq \emptyset$, then clearly $K \cap V_T \neq \emptyset$. Suppose that $K \cap S = \emptyset$. Because $\mathcal{T}_S$ is the unique minimal subtree of $\mathcal{T}_G$ that contains all maximal cliques of G that have at least one vertex of S, we find that K separates two nodes K_1 and K_2 in $\mathcal{T}_S$ for which $K_1 \cap S \neq \emptyset$ and $K_2 \cap S \neq \emptyset$. This means that K separates two vertices $u \in K_1 \cap S$ and $v \in K_2 \cap S$ in G. Since T is a tree and $u, v \in V_T$, at least one vertex of T must be in K. □

Let l denote the number of leaves in $\mathcal{T}_G$; if $\mathcal{T}_G$ consists of one node, then we say that this node is a leaf of $\mathcal{T}_G$.

Lemma 7. *The graph H is a contraction of G if and only if there are pairwise disjoint nonempty sets of vertices $S_1, \ldots, S_k \subseteq V_G$, each of size at most l, such that*

1. *$V_G \subseteq I(S_1) \cup \ldots \cup I(S_k)$;*
2. *$V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}} \neq \emptyset$ if and only if $x_i x_j \in E_H$ for $1 \leq i < j \leq k$;*
3. *G has pairwise vertex-disjoint trees $T_1, \ldots, T_k$ with $S_i \subseteq V_{T_i} \subseteq I(S_i)$ for $1 \leq i \leq k$.*

Proof. First suppose that H is a contraction of G. Consider a corresponding H-witness structure $\mathcal{W}$ of G. For $i = 1, \ldots, k$, let $\mathcal{T}_i$ be the subgraph of $\mathcal{T}_G$ induced by the maximal cliques of G that contain one or more vertices of $W(x_i)$. Because each $W(x_i)$ induces a connected subgraph of G, each $\mathcal{T}_i$ is connected. This means that $\mathcal{T}_i$ is a subtree of $\mathcal{T}_G$, i.e., $\mathcal{T}_i = \mathcal{T}_{W(x_i)}$. We construct S_i as follows. For each leaf K of $\mathcal{T}_i$, we choose a vertex of $W(x_i) \cap K$ and include it in the set S_i. Because $\mathcal{T}_G$ has l leaves, each $\mathcal{T}_i$ has at most l leaves. Hence, $|S_i| \leq l$ for $i = 1, \ldots, k$. We now check conditions 1–3 of the lemma.

1. By construction, we have $\mathcal{T}_i = \mathcal{T}_{S_i}$. All vertices of $W(x_i)$ are inner vertices for $\mathcal{T}_i$, so $W(x_i) \subseteq I(\mathcal{T}_i) = I(\mathcal{T}_{S_i}) = I(S_i)$. Hence, $V_G = \bigcup_{i=1}^{k} W(x_i) \subseteq \bigcup_{i=1}^{k} I(S_i)$.

2. Any two vertices $u, v \in V_G$ are adjacent if and only if there is a maximal clique K in G containing u and v. Hence, two witness sets $W(x_i)$ and $W(x_j)$ are adjacent if and only if there is a maximal clique K in G such that $K \cap W(x_i) \neq \emptyset$ and $K \cap W(x_j) \neq \emptyset$. This means that $W(x_i)$ and $W(x_j)$ are adjacent if and only if $V_{\mathcal{T}_i} \cap V_{\mathcal{T}_j} \neq \emptyset$. It remains to recall that $\mathcal{T}_i = \mathcal{T}_{S_i}$ and $\mathcal{T}_j = \mathcal{T}_{S_j}$, and that two witness sets $W(x_i)$ and $W(x_j)$ are adjacent if and only if $x_i x_j \in E_H$.

3. Every $G[W(x_i)]$ is a connected graph. Hence, every $G[W(x_i)]$ contains a spanning tree T_i. Because the sets $W(x_1), \ldots, W(x_k)$ are pairwise disjoint, the trees T_1, …, T_k are pairwise vertex-disjoint. Moreover, as we already deduced, $S_i \subseteq V_{T_i} = W(x_i) \subseteq I(S_i)$ for $i = 1, \ldots, k$.

Now suppose that there are pairwise disjoint nonempty sets of vertices $S_1, \ldots, S_k \subseteq V_G$, each of size at most l, that satisfy conditions 1–3 of the lemma. By condition 3, there exist pairwise vertex-disjoint trees $T_1, \ldots, T_k$ with $S_i \subseteq V_{T_i} \subseteq I(S_i)$ for $i = 1, \ldots, k$. By condition 1, we have $V_G \subseteq I(S_1) \cup \ldots \cup I(S_k)$. This means that there is a partition $X_1, \ldots, X_k$ of $V_G \setminus \bigcup_{i=1}^{k} V_{T_i}$, where some of the sets X_i can be empty, such that $X_i \subseteq I(S_i)$ for $i = 1, \ldots, k$. Let $W(x_i) = V_{T_i} \cup X_i$ for $i = 1, \ldots, k$. We claim that the sets $W(x_1), \ldots, W(x_k)$ form an H-witness structure of G. By definition, $W(x_1), \ldots, W(x_k)$ are pairwise disjoint, nonempty, and $W(x_1) \cup \ldots \cup W(x_k) = V_G$, i.e., they form a partition of V_G. It remains to show that these sets satisfy conditions (i) and (ii) of the definition of an H-witness structure.

(i) By definition, each tree T_i is connected. By Lemma 6, each node K of $\mathcal{T}_{S_i}$ contains a vertex of T_i. By definition, $X_i \subseteq I(S_i)$, which implies that for each $v \in X_i$, there is a node K in T_{S_i} such that $v \in K$. Because K is a clique in G, we then find that v is adjacent to at least one vertex of T_i. Therefore, each $W(x_i)$ induces a connected subgraph of G.

(ii) For the forward direction, suppose that $W(x_i)$ and $W(x_j)$ are

two adjacent witness sets. Then there exist two vertices $u \in W(x_i)$ and $v \in W(x_j)$ such that $uv \in E_G$. Let K be a maximal clique that contains both u and v. Because $u \in W(x_i)$ and $v \in W(x_j)$, we find that K is a node of $\mathcal{T}_{S_i}$ and of $\mathcal{T}_{S_j}$, respectively. Hence, $V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}} \neq \emptyset$, which means that $x_i x_j \in E_H$ by condition 2.

For the reverse direction, let x_i and x_j be two adjacent vertices in H. By condition 2, we find that $V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}} \neq \emptyset$. Hence, there is a node $K \in V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}}$. By Lemma 6, we deduce that K contains a vertex $u \in V_{T_i}$ and a vertex $v \in V_{T_j}$. Because K is a clique in G, this means that u and v are adjacent. Because $V_{T_i} \subseteq W(x_i)$ and $V_{T_j} \subseteq W(x_j)$, we obtain $u \in W(x_i)$ and $v \in W(x_j)$, respectively. Hence, $W(x_i)$ and $W(x_j)$ are adjacent. □

5.2 The Algorithm

We are now ready to describe our algorithm for Contractibility on chordal graphs.

Theorem 8. *The* Contractibility *problem can be solved in* $n^{O(|V_H|^2)}$ *time on chordal graphs.*

Proof. Let G be a chordal graph on n vertices, and let H be a graph on k vertices with $V_H = \{x_1, \ldots, x_k\}$. If $k > n$ or the number of connected components of G and H are different, then return `No`. Suppose that G and H have $r > 1$ connected components $G_1, \ldots, G_r$ and $H_1, \ldots, H_r$, respectively. For each permutation $\langle i_1, \ldots, i_r \rangle$ of the ordered set $\langle 1, \ldots, r \rangle$, check whether H_{i_j} is a contraction of G_j for every $j \in \{1, \ldots, r\}$. Return `Yes` if this is the case for some permutation, and `No` otherwise. Hence, we may assume that G and H are connected.

Construct a clique tree $\mathcal{T}_G$ of G. If $\mathcal{T}_G$ has at least $k+1$ leaves, then consider each set $A \subseteq V_G$ with $|A| = k$, and continue with $G(A)$ instead of G. This is allowed due to Lemma 5. Note that a clique tree of $G(A)$ has at most $|A| = k$ leaves due to Lemma 4. Hence, we may assume that $\mathcal{T}_G$ has at most k leaves.

Consider each collection of pairwise disjoint sets $S_1, \ldots, S_k$, where each S_i is a subset of V_G with $1 \leq |S_i| \leq k$. For each collection $S_1, \ldots, S_k$, construct the subtrees $\mathcal{T}_{S_1}, \ldots, \mathcal{T}_{S_k}$ of $\mathcal{T}_G$ and the sets $I(S_1), \ldots, I(S_k)$, and test whether conditions 1–3 of Lemma 7 are satisfied. If so, the algorithm returns `Yes`; otherwise, it returns `No`. Correctness of this algorithm is an immediate consequence of Lemma 7.

We now analyze the running time. The number of permutations of the ordered set $\langle 1, \ldots, r \rangle$ is at most $r! \leq k!$. Constructing $\mathcal{T}_G$ takes linear

time. The number of k-element subsets A of V_G is $n^{O(k)}$. The number of collections of sets $S_1, \ldots, S_k$ with $|S_i| \leq k$ for $i = 1, \ldots, k$ is $n^{O(k^2)}$. Constructing subtrees $\mathcal{T}_{S_1}, \ldots, \mathcal{T}_{S_k}$ of $\mathcal{T}_G$ and sets $I(S_1), \ldots, I(S_k)$, and testing if conditions 1–2 of Lemma 7 are satisfied, takes $n^{O(1)}$ time. As a result of Corollary 1, testing whether condition 3 of Lemma 7 is satisfied takes $n^{O(k^2)}$ time. Hence, the total running time is $n^{O(k^2)}$. Consequently, we can test in this running time whether a given chordal graph G contains a given graph H as a contraction. When the graph H, and hence k, is fixed, the running time is polynomial in n. □

The algorithm for Contractibility can be modified for Induced Minor, but it is easier to use the following observation. Let $P_1 \bowtie G$ denote the graph obtained from a graph G by adding a new vertex and making it adjacent to every vertex of G.

Lemma 9 ([22])**.** *Let G and H be two arbitrary graphs. Then G contains H as an induced minor if and only if $(P_1 \bowtie G)$ contains $(P_1 \bowtie H)$ as a contraction.*

Since $P_1 \bowtie G$ is chordal whenever G is chordal, we can combine Lemma 9 with Theorem 8 to obtain the following result, with the same running time as in the proof of Theorem 8.

Corollary 2. Induced Minor *can be solved in $n^{O(|V_H|^2)}$ time on chordal graphs.*

6 Induced Disjoint Paths and Induced Topological Minors

We start this section by showing that the Induced Disjoint Paths problem is polynomial-time solvable on chordal graphs. Recall that Disjoint Paths is NP-complete even on interval graphs [30], and that Induced Disjoint Paths is NP-complete on general graphs already when $k = 2$.

Let G be a graph that, together with a set of terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$, constitutes an instance of Induced Disjoint Paths. We say that a set of paths $P_1, \ldots, P_k$ forms a *solution* of this instance if $P_1, \ldots, P_k$ are mutually induced and P_i is an (s_i, t_i)-path for $i = 1, \ldots, k$. Recall that it is sufficient to consider solutions where each path P_i is an induced path in G. Moreover, two different paths P_i and P_j can only intersect in terminals, and every edge in $G[\bigcup_{i=1}^{k} V(P_i)]$ belongs to some path P_i or

is an edge between two terminal vertices belonging to different terminal pairs.

Theorem 10. *The* Induced Disjoint Paths *problem can be solved in* $O(kn^3)$ *time on chordal graphs.*

Proof. We apply dynamic programming to solve the Induced Disjoint Paths problem on chordal graphs. Because our approach is similar to the one we used for Set-Restricted Disjoint Paths, we mainly explain the differences.

Let G be a chordal graph that together with terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ forms an instance of Induced Disjoint Paths. If G is disconnected, then we check for each pair of terminals (s_i, t_i) whether s_i and t_i belong to the same connected component. If not, then we return `No`. Otherwise, we consider each connected component and its set of terminals separately. Hence, we may assume that G is connected.

As before, we construct in linear time a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ of G with root X_r, such that each bag is a clique in G. We then apply our dynamic programming algorithm on $(\mathcal{T}, \mathcal{X})$. Recall that for a node $X_i \in V_{\mathcal{T}}$, we let $\mathcal{T}_i$ denote the subtree of $\mathcal{T}$ with root X_i that is induced by X_i and all its descendants, and we write $G_i = G[\bigcup_{j \in V_{\mathcal{T}_i}} X_j]$.

For each node X_i, we construct a table that stores a collection of records

$$\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k)),$$

where $R_1, \ldots, R_k \subseteq X_i$ are ordered sets without common vertices except (possibly) terminals and $0 \leq |R_j| \leq 2$ for $j \in \{1, \ldots, k\}$, and where each $State_j$ can have one of the following four values:

Not initialized, *Started from* s, *Started from* t, or *Completed*.

These records correspond to the partial solution of Induced Disjoint Paths for G_i with exactly the same properties as the records stored in the tables for Set-Restricted Disjoint Paths. In particular, each R_j is the set of vertices of the (s_j, t_j)-path in X_i.

The tables are constructed and updated in a straightforward way using the following two observations: (i) every induced path contains at most two vertices of each clique, and (ii) every clique contains internal vertices of at most one path. The algorithm computes these tables for all nodes of $\mathcal{T}$, starting from the leaves. Finally, the table for the root X_r is constructed. The algorithm returns `Yes` if the table for X_r contains a record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ with $State_1 = \ldots = State_k = Completed$, and it returns `No` otherwise.

For the running time, it is sufficient to note that by observations (i) and (ii), there are at most kn^2 possibilities to include in a partial solution internal vertices of the paths that go through the vertices of X_i. Because $\mathcal{T}$ contains $O(n)$ nodes, this means that the total running time is $O(kn^3)$. □

Corollary 3. Induced Topological Minor *can be solved in* $O(|E_H| \cdot n^{|V_H|+3})$ *time on chordal graphs.*

Proof. Let G be a chordal graph on n vertices and H be a graph whose vertices are ordered as $x_1, \ldots, x_{|V_H|}$. Recall that G contains H as an induced topological minor if and only if G contains an induced subgraph isomorphic to a subdivision of H. In G we choose $|V_H|$ vertices that we order, say as $u_1, \ldots, u_{|V_H|}$. For each edge $x_ix_j \in E_H$ we define a terminal pair (u_i, u_j). This leads to a set of terminal pairs $T = \{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}$ where $\ell = |E_H|$. Then G contains an induced subgraph isomorphic to a subdivision of H such that the isomorphism maps u_i to x_i for $i = 1, \ldots, |V_H|$ if and only if G contains a set of ℓ mutually induced paths $P_1, \ldots, P_\ell$, such that P_j has end-vertices s_j and t_j for $j = 1, \ldots, \ell$. Hence, we may apply Theorem 10 to solve the latter problem in $O(\ell n^3)$ time. If this does not yield a solution, then we choose another ordered set of u-vertices until we have considered them all. Because the number of such choices is $O(n^{|V_H|})$ and $\ell = |E_H|$, the result follows. □

7 Concluding Remarks

In this section, we give another application of the Set-Restricted Disjoint Paths problem. We show that Set-Restricted Disjoint Connected Subgraphs can be solved in polynomial time on interval graphs for every fixed k. A graph is an *interval graph* if intervals of the real line can be associated with its vertices in such a way that two vertices are adjacent if and only if their corresponding intervals overlap. Interval graphs constitute another important subclass of chordal graphs besides split graphs.

Proposition 11. *The* Set-Restricted Disjoint Connected Subgraphs *problem can be solved in* $n^{O(k)}$ *time on interval graphs.*

Proof. Let $G = (V, E)$ be an interval graph that together with k pairwise disjoint nonempty vertex subsets $S_1, \ldots, S_k$ with corresponding domains $U_1, \ldots, U_k$ for some $k \geq 1$ forms an instance of Set-Restricted Disjoint Connected Subgraphs.

We construct an interval representation of G; this can be done in linear time as shown by Booth and Lueker [5]. Given this representation, we say that a vertex u in a subset $V' \subseteq V$ is a *leftmost* vertex of V' if there is no vertex in V' whose associated interval contains a point placed on the real line before the interval of u starts. We define the notion of a *rightmost* vertex analogously. For $i = 1, \ldots, k$, let s_i and t_i be a leftmost and rightmost vertex of S_i, respectively. We define $U_i' = U_i \setminus \bigcup_{h \neq i} S_h$ for $i = 1, \ldots, k$.

We make the following observation. Let $1 \leq i \leq k$, and let P be an arbitrary path from s_i to t_i. Then, by our choice of s_i and t_i, we find that P *dominates* S_i, i.e., every vertex of S_i that is not on P is adjacent to a vertex of P. Moreover, our choice of s_i and t_i also implies that every connected subgraph that contains S_i contains a path from s_i to t_i that dominates S_i. Hence, the terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ with domains $U_1', \ldots, U_k'$, respectively, form an instance of Set-Restricted Disjoint Paths that is a `Yes`-instance of this problem if and only if $(G, S_1, \ldots, S_k, U_1, \ldots, U_k)$ is a `Yes`-instance of Set-Restricted Disjoint Connected Subgraphs. Because we can solve the first problem by Theorem 3 in $n^{O(k)}$ time, the result follows. □

We conclude with some open questions. Recall that Contractibility and Induced Minor are W[1]-hard with parameter $|V_H|$ on chordal graphs [19]. Is either of these problems in FPT on interval graphs? Is Set-Restricted Disjoint Connected Subgraphs in FPT with parameter $|S_1| + |S_2| + \cdots + |S_k|$ on chordal graphs? Even though an affirmative answer to the last question would not improve our results for Contractibility, Induced Minor and Induced Topological Minor, the question might be interesting in its own right.

References

[1] R. Belmonte, P. A. Golovach, P. Heggernes, P. van 't Hof, M. Kamiński, and D. Paulusma. Finding contractions and induced minors in chordal graphs via disjoint paths. In: *Proceedings of ISAAC 2011*, LNCS 7074: 110–119, Springer.

[2] R. Belmonte, P. Heggernes, and P. van 't Hof. Edge contractions in subclasses of chordal graphs. In: *Proceedings of TAMC 2011*, LNCS 6648: 528–539, Springer, 2011.

[3] D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90: 85–92, 1991. See also Corrigendum. *Discrete Mathematics*, 102: 109, 1992.

[4] J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. In *Graph Theory and Sparse Matrix Computations*, IMA Volumes in Mathematics and its Applications 56: 1–29, Springer 1993.

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

[6] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey.* SIAM, 1999.

[7] A. E. Brouwer and H. J.Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11: 71–79, 1987.

[8] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[9] G. A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25: 71–76, 1961.

[10] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141:109–131, 1995.

[11] J. Fiala, M. Kamiński, B. Lidický, and D. Paulusma. The k-in-a-path problem for claw-free graphs. *Algorithmica*, 62: 499-519, 2012.

[12] J. Fiala, M. Kamiński and D. Paulusma. A note on contracting claw-free graphs. Manuscript, 2011.

[13] M. R. Fellows, J. Kratochvíl, M. Middendorf, and F. Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13: 266–282, 1995.

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

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

[17] A. George, and J. W Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall Professional Technical Reference (1981)

[18] P. A. Golovach, M. Kamiński, and D. Paulusma, Contracting a chordal graph to a split graph or a tree. In: *Proceedings of MFCS 2011*, LNCS 6907: 339–350, Springer, 2011.

[19] P. A. Golovach, M. Kamiński, D. Paulusma and D. M. Thilikos. Containment relations in split graphs. *Discrete Applied Mathematics*, 160: 155–163, 2012.

[20] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics 57, Elsevier (2004)

[21] M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In: *Proceedings of STOC 2011*, 479–488.

[22] P. van 't Hof, M. Kamiński, D. Paulusma, S. Szeider and D. M. Thilikos. On graph contractions and induced minors. *Discrete Applied Mathematics*, 160: 799–809, 2012.

[23] P. van 't Hof, D. Paulusma,and G. J. Woeginger. Partitioning graphs in connected parts. *Theoretical Computer Science*, 410: 4834–4843, 2009.

[24] M. Kamiński and D. M. Thilikos. Contraction checking in graphs on surfaces. In: *Proceedings of STACS 2012*, to appear.

[25] T. Kloks. *Treewidth, Computations and Approximations.* LNCS 842, Springer, 1994.

[26] B. Lévêque, D. Y. Lin, F. Maffray, and N. Trotignon. Detecting induced subgraphs. *Discrete Applied Mathematics*, 157: 3540–3551, 2009.

[27] A. Levin, D. Paulusma, and G. J. Woeginger. The computational complexity of graph contractions I: polynomially solvable and NP-complete cases. *Networks*, 51: 178–189, 2008.

[28] A. Levin, D. Paulusma, and G. J. Woeginger. The computational complexity of graph contractions II: two tough polynomially solvable cases. *Networks* 52: 32–56, 2008.

[29] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108: 343–364, 1992.

[30] S. Natarajan and A. P. Sprague. Disjoint paths in circular arc graphs. *Nordic Journal on Computing* 3: 256–270, 1996.

[31] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.

[32] H. N. de Ridder et al. Information System on Graph Classes and their Inclusions (ISGCI). `http://www.graphclasses.org`, 2001-2012.

[33] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63: 65–110, 1995.

[34] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press graduate series Mathematics and its Applications, 2003.

[35] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13: 66–579, 1984.

Chapter 8

Induced Immersions

Induced Immersions

Rémy Belmonte[1] Pim van 't Hof[1]
Marcin Kamiński[2,]

[1] Department of Informatics, University of Bergen, Norway
{remy.belmonte,pim.vanthof}@ii.uib.no
[2] Département d'Informatique, Université Libre de Bruxelles, Belgium
Marcin.Kaminski@ulb.ac.be

Abstract

A graph G contains a multigraph H as an induced immersion if H can be obtained from G by a sequence of vertex deletions and lifts. We present a polynomial-time algorithm that decides for any fixed multigraph H whether an input graph G contains H as an induced immersion. We also show that for every multigraph H with maximum degree at most 2, there exists a constant c_H such that every graph with treewidth more than c_H contains H as an induced immersion.

1 Introduction

A recurrent problem in algorithmic graph theory is to decide, given two graphs G and H, whether the structure of H appears as a pattern within the structure of G. The notion of appearing as a pattern gives rise to various graph containment problems depending on which operations are allowed. Maybe the most famous example is the minor relation that has been widely studied, in particular in the seminal Graph Minor series of papers by Robertson and Seymour (see, e.g., [16, 17]). A graph G contains a graph H as a *minor* if H can be obtained from G by a sequence of vertex deletions, edge deletions and edge contractions. One of the highlights of the Graph Minor series is the proof that, for every fixed graph H, there exists a cubic-time algorithm that decides whether an input graph G contains H as a minor [16].

If the contraction operation is restricted so that we may only contract an edge if at least one of its endpoints has degree 2, then we obtain the

operation known as *vertex dissolution*. If H can be obtained from G by a sequence of vertex deletions, edge deletions and vertex dissolutions, then H is a *topological minor* of G. It is a trivial observation that if G contains H as topological minor, then it also contains H as a minor. Numerous results on topological minors exist in the literature, notable examples being the recent proof that testing for topological minors is FPT by Grohe et al. [13] and the characterization of graphs excluding a fixed graph as a topological minor by Grohe and Marx [14].

Grohe and Marx [14] also show that another containment relation can be decided in FPT time, namely *immersion*. In order to define immersion, we first need to define another graph operation. The *lift* (or *split-off*) operation is defined as follows: given three (not necessarily distinct) vertices u, v, w such that $uv, vw \in E(G)$, we delete uv and vw and replace them by a new edge uw, possibly creating a loop or multiple edges. A graph G contains H as an *immersion* if H can be obtained from G by a sequence of vertex deletions, edge deletions, and lifts. It is not hard to verify that if G contains H as a topological minor, then G also contains H as an immersion, as a dissolution can be simulated by a lift and a vertex deletion. The interest in the immersion relation has steadily been growing [2, 3, 9, 10, 12, 13, 18, 20].

1.0.1 Our results.

We introduce and study the *induced immersion* relation, which is equivalent to the immersion relation where no edge deletions are allowed (see Figure 1 for an overview of the different containment relations mentioned in this paper). Our main result is that, for any fixed multigraph H, there exists a polynomial-time algorithm deciding whether a simple graph G contains H as an induced immersion. It is interesting to note that it is highly unlikely that a similar result exists for the induced minor and induced topological minor relations, as there exist fixed graphs H for which the problem of deciding whether a graph contains H as an induced minor or as an induced topological minor is NP-complete [8, 15]. We complement this result by showing that, for every fixed multigraph H of maximum degree at most 2, there exists a constant c_H such that every graph with treewidth more than c_H contains H as an induced immersion.

2 Preliminaries

Following the terminology of Diestel [4], we define a *multigraph* to be a pair $G = (V, E)$, where V is a set, and E is a multiset such that every

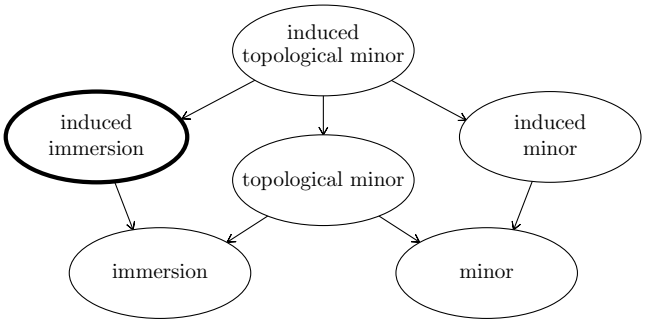


Figure 1: The relationship between the different containment relations mentioned in this paper. An arrow from relation A to relation B indicates that if G contains H with respect to relation A, then G also contains H with respect to relation B. For example, if G contains H as an induced topological minor, then G also contains H as an induced immersion, since a dissolution can be simulated by a lift and a vertex deletion.

$e \in E$ is a multiset of two elements of V. The elements of V and E are called the *vertices* and *edges* of G, respectively. For convenience, we write uv instead of $\{u, v\}$ to denote an edge between u and v; in particular, uu denotes a loop at u. The *multiplicity* of an edge $uv \in E$, denoted by $\text{mult}_G(uv)$, is the number of times the element $\{u, v\}$ appears in the multiset E. By slight abuse of notation, we define $\text{mult}(uv) = 0$ whenever $uv \notin E$. A *graph* is a multigraph without loops in which the multiplicity of every edge is at most 1. We refer to the textbook by Diestel [4] for graph terminology not defined below, and to the monograph by Downey and Fellows [6] for a background on parameterized complexity.

Let $G = (V, E)$ be a multigraph. For two vertices $u, v \in V$, we say that u is a *neighbor* of v if $\{u, v\} \in E$. Note that if $\{u, u\} \in E$, then u is its own neighbor. The *degree* of a vertex $u \in V$, denoted by $d_G(u)$, is the number of neighbors of u, where a loop at u contributes 2 to the degree of u. Let P be a path in G from u to v that has at least one edge. The *length* of P is its number of edges. The vertices u and v are the *endpoints* of P, and every other vertex of P is an *internal* vertex of P. By abuse of terminology, we will allow the endpoints u and v of a path to be the same vertex, even though strictly speaking such a path is a cycle.

Let G be a multigraph. Let u, v, w be three (not necessarily distinct) vertices in G such that $uv, vw \in E(G)$. The *lift* (or *split-off*) $\{uv, vw\}$ of the edges uv and vw is the operation that deletes uv and vw and replaces them by a new edge uw, thereby increasing the multiplicity of uv by exactly 1. In particular, note that lifting two copies of an edge uv

creates a loop either at u or at v, and lifting a loop uu with any incident edge uv simply deletes the loop uu and leaves uv unchanged. When we lift two edges uv and vw, we say that v is the *pivot* of the lift $\{uv, vw\}$. The multigraph obtained from G by applying the lift $\{uv, vw\}$ is denoted by $G \vee \{uv, vw\}$. Similarly, given a sequence of lifts $\mathcal{L} = (\ell_1, \ldots, \ell_q)$, we define $G \vee \mathcal{L} = (((G \vee \ell_1) \vee \ell_2) \cdots \vee \ell_q)$. Let $P = p_1 \cdots p_\ell$ be a path of length at least 2 in G. When we say that we *lift* the path P, we mean that we lift the edges x_1x_i and x_ix_{i+1} for i from 2 up to $p-1$. Note that lifting the path P is equivalent to deleting all the edges of P from G, and adding a new edge with endpoints $\phi(u)\phi(v)$.

Let G and H be two multigraphs. Then G contains H as an *immersion* if H can be obtained from G by a sequence of vertex deletions, edge deletions, and lifts. It is known that G contains H as an immersion if and only if there exists a subset S of $|V(H)|$ vertices in G and a bijection ϕ from $V(H)$ to S such that, for each edge $uv \in E(H)$, there exists a path P_{uv} in G from $\phi(u)$ to $\phi(v)$, and all these paths P_{uv}, $uv \in E(H)$, are mutually edge-disjoint. We point out that if $uu \in E(H)$, then the path P_{uu} starts and ends in the same vertex; even though strictly speaking P_{uu} is a cycle, recall that P_{uu} satisfies our definition of a path (from u to u). If H can be obtained from G by a sequence of vertex deletions and lifts, then G contains H as an *induced immersion*. Since we may assume that in such a sequence the lifts appear before any vertex deletion, we can make the following observation.

Observation 1. *Let G and H be two multigraphs. Then G contains H as an induced immersion if and only if there exists a sequence of lifts $\mathcal{L}$ such that $G \vee \mathcal{L}$ contains H as an induced subgraph.*

The definition of induced immersion gives rise to the following definition, which will be used frequently throughout the paper.

Definition 1 (H-model)**.** *Let G and H be two multigraphs such that G contains H as an induced immersion. Given a sequence of lifts $\mathcal{L}$, a set of vertices $S \subseteq V(G)$ and a bijection ϕ from $V(H)$ to S, we say that $(S, \mathcal{L}, \phi)$ is an* H-model *of G if ϕ is an isomorphism from H to $G'[S]$, where $G' = G \vee \mathcal{L}$. The vertices of S are referred to as the* branch vertices *of the H-model $(S, \mathcal{L}, \phi)$.*

Observation 2. *Let G and H be two multigraphs. Then G contains H as an induced immersion if and only if G has an H-model.*

The Induced Immersion problem takes as input a graph G and a multigraph H, and the task is to decide whether G contains H as an induced immersion.

The *elementary wall* of height r is the graph W_r whose vertex set is $\{(x,y) \mid 0 \leq x \leq 2r+1, 0 \leq y \leq r\} \setminus \{(0,0),(2r+1,r)\}$ if r is even and $\{(x,y) \mid 0 \leq x \leq 2r+1, 0 \leq y \leq r\} \setminus \{(0,0),(0,r)\}$ if r is odd, and such that there is an edge between any vertices (x,y) and (x',y') if either $|x'-x| = 1$ and $y = y'$, or if $x = x', |y'-y| = 1$ and x and $\max\{y,y'\}$ have the same parity. The elementary walls of height 2, 3 and 4 are depicted in Figure 2.

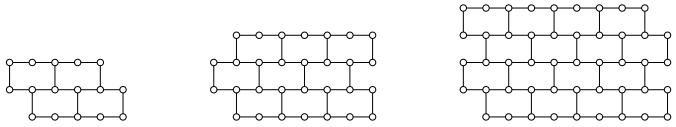


Figure 2: Elementary walls of height 2, 3, and 4.

Robertson, Seymour and Thomas [19] proved that for $r \geq 1$, every graph with treewidth more than 20^{2r^5} contains the $r \times r$-grid as a minor. Note that every $r \times r$-grid contains an elementary wall of height $\lfloor r/2 \rfloor - 1$ as a minor. Since an elementary wall has maximum degree at most 3, and every minor with maximum degree at most 3 is also a topological minor of the same graph, every $r \times r$-grid also contains $W_{\lfloor r/2 \rfloor - 1}$ as a topological minor. As we pointed out in the introduction, this means that $W_{\lfloor r/2 \rfloor - 1}$ is also contained as an immersion in an $r \times r$-grid. Hence, the aforementioned result of Robertson, Seymour and Thomas implies the following result.

Theorem 1 ([19]). *For $r \geq 1$, every graph with treewidth more than $20^{10(r+1)^5}$ contains W_r as an immersion.*

3 Finding a fixed multigraph H as an induced immersion

The goal of this section is to show that, for every fixed multigraph H, there exists a polynomial-time algorithm that decides whether a graph G contains H as an induced immersion. We first show that if both G and H are given as input, this problem cannot be solved in polynomial time, unless P = NP.

Lemma 1. *The* Induced Immersion *problem is NP-complete, even if G and H are disjoint unions of paths.*

Proof. We reduce from the INDUCED SUBGRAPH problem on on disjoint unions of paths. The problem was observed to be NP-complete by Damaschke [1], even where the graphs G and H do not contain isolated vertices. We claim that given two disjoint unions of paths G and H without isolated vertices, G contains H as an induced subgraph if and only if G contains H as an induced immersion. The forward direction is obvious from the definition of induced subgraph and induced immersion. For the backward direction, we claim that for every pair of incident edges uv, vw of G, there is a vertex x such that $G \vee \{uv, vw\}$ and $G - x$ are isomorphic. This follows from the fact that H does not contain isolated vertices, and the vertex v is isolated in $G \vee \{uv, vw\}$, hence v must be deleted from $G \vee \{uv, vw\}$ in order to obtain a graph isomorphic to H. If we take x to be any of the two endpoints of the path of G that contains u, v and w, then we can easily observe that $G - x$ is isomorphic to $(G \vee \{uv, vw\}) - v$. Thus we conclude that G contains H as an induced subgraph if and only if G contains H as an induced immersion, which implies that INDUCED IMMERSION is NP-complete even when G and H are disjoint unions of paths. □

Throughout this section, let G and H be two multigraphs such that G contains H as an induced immersion, and let $(S, \mathcal{L}, \phi)$ be an H-model of G. Lemma 3 below shows that we may assume $\mathcal{L}$ to satisfy a property that will be exploited in the algorithm in the proof of Theorem 2. The proof of Lemma 3 relies on the following result, which states that we can "safely" swap certain consecutive pairs of lifts in $\mathcal{L}$.

Lemma 2. *Let G and H be two multigraphs such that G contains H as an induced immersion, and let $(S, \mathcal{L}, \phi)$ be an H-model of G. Let $\{xy, yz\}$ and $\{uv, vw\}$ be two consecutive lifts in $\mathcal{L}$, occuring at positions $i-1$ and i in $\mathcal{L}$, respectively, such that $\{xy, yz\} \notin \{\{uy, yv\}, \{vy, yw\}\}$. Then G has an H-model $(S, \mathcal{L}', \phi)$ such that $\mathcal{L}'$ can be obtained from $\mathcal{L}$ by swapping $\{xy, yz\}$ and $\{uv, vw\}$.*

Proof. Let us denote by $\mathcal{L}^-$ the sequence of lifts containing the first $i-2$ elements of $\mathcal{L}$ in the same order, and let $G^- = G \vee \mathcal{L}^-$. First, observe that we have $xy, yz \in E(G^-)$ by the definition of $\mathcal{L}$ and G^-. Moreover, both the edges uv and vw must be present in G^-, since $\{xy, yz\} \notin \{\{uy, yv\}, \{vy, yw\}\}$. Therefore, we can apply the lift $\{uv, vw\}$ in G^-. It is clear that $xy, yz \in E(G^- \vee \{uv, vw\})$, which means that we can apply the lift $\{xy, yz\}$ in $G^- \vee \{uv, vw\}$. Finally, observe that both $(G^- \vee \{uv, vw\}) \vee \{xy, yz\}$ and $(G^- \vee \{xy, yz\}) \vee \{uv, vw\}$ are isomorphic to $G^* = (V(G), E(G^-) \setminus \{uv, vw, xy, yz\} \cup \{uw, xz\})$. Hence $(G^- \vee$

$\{uv, vw\}) \vee \{xy, yz\}$ and $(G^- \vee \{xy, yz\}) \vee \{uv, vw\}$ are isomorphic, and the lemma follows. □

Lemma 3. *Let G and H be two multigraphs such that G contains H as an induced immersion, and let $(S, \mathcal{L}, \phi)$ be an H-model of G. For any edge uv of $G[S]$, there exists an H-model $(S, \mathcal{L}', \phi)$ of G such that uv appears in the first lift in $\mathcal{L}'$.*

Proof. Out of all the sequences of lifts $\mathcal{L}$ such that G has an H-model $(S, \mathcal{L}, \phi)$, let $\mathcal{L}'$ be one such that the position of the first lift in $\mathcal{L}'$ that contains the edge uv is as small as possible. We claim that uv appears in the first lift of $\mathcal{L}'$. For contradiction, assume that the first lift $\ell \in \mathcal{L}$ that contains uv occurs at position $i \geq 2$ in $\mathcal{L}'$. Without loss of generality, let $\ell = \{uv, vw\}$ for some vertex w of G. Let $\ell' = \{xy, yz\}$ be the lift occuring at position $i-1$ in $\mathcal{L}'$, and let G^- be the graph obtained from G by applying the first $i-2$ lifts of $\mathcal{L}'$.

First suppose that $\{xy, yz\} \notin \{\{uy, yv\}, \{vy, yw\}\}$. Then we know from Lemma 2 that there exists an H-model $(S, \mathcal{L}'', \phi)$ of G such that $\{uv, vw\}$ occurs before $\{xy, yz\}$ in $\mathcal{L}''$. This contradicts the choice of $\mathcal{L}'$.

Now suppose $\{xy, yz\} \in \{\{uy, yv\}, \{vy, yw\}\}$. First consider the case where $\ell' = \{uy, yv\}$. Note that the edge uv is present in G^-, as we assumed that uv is an edge of $G[S]$ and ℓ is the first lift in $\mathcal{L}'$ that contains uv. It is clear that also the edges vw, uy and yv are present in G^-. Hence, we can apply exactly the same arguments as in the proof of Lemma 2 to show that G has an H-model $(S, \mathcal{L}'', \phi)$ where $\mathcal{L}''$ is obtained from $\mathcal{L}'$ by swapping ℓ and ℓ'. This again contradicts the choice of $\mathcal{L}'$.

Finally, consider the case where $\ell' = \{vy, yw\}$. We claim that we can replace the lifts $\{vy, yw\}$ and $\{uv, vw\}$ in $\mathcal{L}'$ by the lifts $\{uv, vy\}$ and $\{uy, yw\}$ and obtain a new sequence of lifts $\mathcal{L}''$ such that $(S, \mathcal{L}'', \phi)$ is an H-model of G. In order to see this, first observe that both the edges uv and vy are present in G^-, which means that we can indeed apply the lift $\{uv, vy\}$ in G^-. Let $G^+ = G^- \vee \{uv, vy\}$. As we lifted the edges uv and vy, G^+ contains the edge uy. Moreover, as the edge yw was already present in G^- and was not lifted, it is also present in G^+. Hence we can apply the lift $\{uy, yw\}$ in G^+. Finally, observe that the graph $(G^- \vee \{vy, yw\}) \vee \{uv, vw\}$ is isomorphic to the graph $(G^- \vee \{uv, vy\}) \vee \{uy, yw\}$. We conclude that $(S, \mathcal{L}'', \phi)$ is indeed an H-model of G. Since uv appears in a lift at position $i-1$ in $\mathcal{L}''$, this contradicts the choice of $\mathcal{L}'$. □

Informally speaking, the next lemma shows that given the H-model $(S, \mathcal{L}, \phi)$ of G, there exists a short sequence of lifts $\mathcal{L}^*$ whose application removes any unwanted edges between vertices of S. Here, an edge between

vertices $\phi(u), \phi(v) \in S$ is unwanted if the multiplicity of $\phi(u)\phi(v)$ in G is strictly larger than the multiplicity of uv in H.

Lemma 4. *Let G be a graph and let H be a multigraph such that G contains H as an induced immersion, and let $(S, \mathcal{L}, \phi)$ be an H-model of G. Then there exists a sequence $\mathcal{L}^*$ of lifts that satisfies the following four properties:*

(i) $|\mathcal{L}^*| \leq |E(G[S])|$;

(ii) for every $\{uv, vw\} \in \mathcal{L}^*$, *we have* $v \in S$ *and* $\{u, w\} \cap S \neq \emptyset$;

(iii) for every $u, v \in V(H)$, *we have* $\mathrm{mult}_{G \vee \mathcal{L}^*}(\phi(u)\phi(v)) \leq \mathrm{mult}_H(uv)$;

(iv) $G \vee \mathcal{L}^*$ *contains* H *as an induced immersion.*

Proof. We first show that condition (ii) implies condition (i). Suppose $\mathcal{L}^*$ is a sequence of lifts that satisfies (ii). Note that applying a lift $\{uv, vw\}$ with $v \in S$ and $\{u, w\} \cap S \neq \emptyset$ decreases the number of edges in $G[S]$ by exactly 1. Hence, after applying exactly $|E(G[S])|$ such lifts, there are no edges left between vertices in S. Then we cannot apply any other lift $\{uv, vw\}$ with $v \in S$ and $\{u, w\} \cap S \neq \emptyset$, implying that $\mathcal{L}^*$ contains at most $|E(G[S])|$ lifts.

Starting from the empty sequence, we will now construct a sequence of lifts $\mathcal{L}^*$ that satisfies conditions (i)–(iv). Note that the empty sequence trivially satisfies (ii), and consequently also satisfies (i). It also satisfies (iv) by the definition of an H-model. If the graph G satisfies $\mathrm{mult}_G(\phi(u)\phi(v)) \leq \mathrm{mult}_H(uv)$ for every $u, v \in V(H)$, then the empty sequence also satisfies (iii), and we are done.

Suppose the empty sequence does not satisfy (iii). Then there exist $u, v \in V(H)$ such that $\mathrm{mult}_{G \vee \mathcal{L}^*}(\phi(u)\phi(v)) > \mathrm{mult}_H(uv)$. Let e be an edge in $G[S]$ between $\phi(u)$ and $\phi(v)$. Due to Lemma 3, there exists an H-model $(S, \mathcal{L}'', \phi)$ of G such that the edge e is contained in the first lift ℓ in the sequence $\mathcal{L}''$. We redefine $\mathcal{L}^*$ to be the sequence obtained from $\mathcal{L}^*$ by adding ℓ as the last lift. Note that this new $\mathcal{L}^*$ still satisfies (ii), as both endpoints of e belong to S. Moreover, $\mathcal{L}^*$ satisfies (iv) by Lemma 3, as we can take $\mathcal{L}'$ to be the sequence obtained from $\mathcal{L}''$ by removing ℓ. If $\mathcal{L}^*$ satisfies (iii), then we are done. Suppose $\mathcal{L}^*$ does not satisfy (iii). If $|\mathcal{L}^*| = |E(G[S])|$, then there are no edges between any two vertices of S in the graph $G \vee \mathcal{L}^*$. Then $\mathrm{mult}_{G \vee \mathcal{L}^*}(\phi(u)\phi(v)) = 0$ for every $u, v \in V(H)$, and thus $\mathcal{L}^*$ satisfies (iii). Since we assumed this not to be the case, we must have $|\mathcal{L}^*| < |E(G[S])|$, and we can repeat the above procedure. Note that this procedure will yield the desired sequence $\mathcal{L}^*$ after at most $|E(G[S])|$ steps. □

The next lemma shows that after unwanted edges between vertices of S have been removed by applying $\mathcal{L}^*$, deciding whether G contains H as an induced immersion is equivalent to finding a family of mutually edge-disjoint paths.

Lemma 5. *Let G and H be two multigraphs. Suppose there is a set $S \subseteq V(G)$ and a bijection ϕ from $V(H)$ to S such that $\text{mult}_G(\phi(u)\phi(v)) \leq \text{mult}_H(uv)$ for every $u, v \in V(H)$. Then G contains H as an induced immersion if and only if the following two properties are satisfied:*

(i) for every $u, v \in V(H)$, there is a set $\mathcal{P}_{uv}$ of $\text{mult}_H(uv)$ paths in G from $\phi(u)$ to $\phi(v)$;

(ii) the paths in $\mathcal{P} = \bigcup_{u,v \in V(H)} \mathcal{P}_{uv}$ are mutually edge-disjoint.

Proof. First suppose that G contains H as an induced immersion. Then G also contains H as an immersion. The existence of the family $\mathcal{P}$ then follows from the definition of immersion.

Now suppose that there exists a family $\mathcal{P}$ of paths in G as mentioned in the lemma. Among all such families $\mathcal{P}$, let $\mathcal{P}' = \bigcup_{u,v \in V(H)} \mathcal{P}'_{uv}$ be one that contains the largest number of paths of length 1.

Claim 1. For every $\phi(u), \phi(v) \in V(G)$, each of the $\text{mult}_G(\phi(u)\phi(v))$ edges between $\phi(u)$ and $\phi(v)$ is the unique edge of a path of length 1 in $\mathcal{P}'_{uv}$.

Proof of Claim 1. For contradiction, suppose there exist $\phi(u), \phi(v) \in V(G)$ such that there is an edge e in G between $\phi(u)$ and $\phi(v)$ that is not the unique edge of a path in $\mathcal{P}'_{uv}$. Note that this means that e is not contained in any path of $\mathcal{P}'_{uv}$. We claim that $\mathcal{P}'_{uv}$ contains a path of length at least 2. For contradiction, suppose that each of the $\text{mult}_H(uv)$ paths in $\mathcal{P}_{uv}$ has length 1. By assumption, $\text{mult}_G(\phi(u)\phi(v)) \leq \text{mult}_H(uv)$. Since all the paths in $\mathcal{P}'$ are edge-disjoint and e is not contained in any path of $\mathcal{P}'_{uv}$, the number of paths in $\mathcal{P}'_{uv}$ can be at most $\text{mult}_H(uv) - 1$, yielding the desired contradiction.

We now distinguish two cases, depending on whether or not e belongs to some path in $\mathcal{P}'$. We obtain a contradiction in both cases, which will imply the validity of the claim.

First suppose that there exists a path $P \in \mathcal{P}'$ such that e is an edge of P. Observe that P must have length at least 2, and hence $P \in \mathcal{P}'_{xy}$ for some $\{x, y\} \neq \{u, v\}$. Recall that $\mathcal{P}'_{uv}$ contains a path P' from $\phi(u)$ to $\phi(v)$ of length at least 2. Consider the two paths P' and P. Recall that P is a path between $\phi(x)$ and $\phi(y)$ containing the edge e. Let P_{uv} be the path that has e as its only edge, and let P_{xy} be the path obtained from P by removing e and adding the internal vertices and the edges of P_{uv}. Observe

that the paths P_{uv} and P_{xy} use exactly the same vertices and edges as P' and P. Let $\mathcal{P}''_{uv} = \mathcal{P}'_{uv} \setminus \{P'\} \cup \{P_{uv}\}$ and $\mathcal{P}''_{xy} = \mathcal{P}'_{xy} \setminus \{P\} \cup \{P_{xy}\}$, and let $\mathcal{P}''_{ab} = \mathcal{P}'_{ab}$ for every $\{a, b\} \notin \{\{u, v\}, \{x, y\}\}$. Then $\mathcal{P}'' = \bigcup_{u,v \in V(H)} \mathcal{P}''_{uv}$ is a family of paths that satisfies properties (i) and (ii), and contains one more path of length 1 than $\mathcal{P}'$, contradicting the choice of $\mathcal{P}'$.

Now suppose that e is not contained in any path of $\mathcal{P}'$. Let $P' \in \mathcal{P}'_{uv}$ be a path of length at least 2, and let P_{uv} be the path that has e as its unique edge. We define $\mathcal{P}''_{uv} = \mathcal{P}'_{uv} \setminus \{P'\} \cup \{P_{uv}\}$ and $\mathcal{P}''_{xy} = \mathcal{P}'_{xy}$ for every $\{x, y\} \neq \{u, v\}$. Then $\mathcal{P}'' = \bigcup_{u,v \in V(H)} \mathcal{P}''_{uv}$ is a family of paths that satisfies properties (i) and (ii), and contains one more path of length 1 than $\mathcal{P}'$. As in the previous case, this contradicts the choice of $\mathcal{P}'$. This concludes the proof of Claim 1. $\diamond$

Consider the set $\mathcal{P}'_{uv}$ for some $u, v \in V(H)$. By property (i), $\mathcal{P}'_{uv}$ consists of $\text{mult}_H(uv)$ mutually edge-disjoint paths in G from $\phi(u)$ to $\phi(v)$. By Claim 1, at least $\text{mult}_G(\phi(u)\phi(v))$ many of these paths have length 1. By the definition of multiplicity, there are exactly $\text{mult}_G(\phi(u)\phi(v))$ edges between $\phi(u)$ and $\phi(v)$, which implies that the set $\mathcal{P}'_{uv}$ contains exactly $\text{mult}_G(\phi(u)\phi(v))$ paths of length 1. This, together with property (ii), implies that no path in $\mathcal{P}' \setminus \mathcal{P}'_{uv}$ contains an edge between $\phi(u)$ and $\phi(v)$. By symmetry, no path in $\mathcal{P}'_{uv}$ contains an edge between $\phi(x)$ and $\phi(y)$ for $\{x, y\} \neq \{u, v\}$. Hence, if we lift each of the $\text{mult}_H(uv) - \text{mult}_G(\phi(u)\phi(v))$ paths in $\mathcal{P}'_{uv}$ of length at least 2, then we create $\text{mult}_H(uv) - \text{mult}_G(\phi(u)\phi(v))$ new edges between $\phi(u)$ and $\phi(v)$, and do not change $\text{mult}_G(\phi(x)\phi(y))$ for any $\{x, y\} \neq \{u, v\}$. Note that the number of edges between $\phi(u)$ and $\phi(v)$ after lifting the paths in $\mathcal{P}'_{uv}$ is exactly $\text{mult}_H(uv)$.

From the above arguments, it is clear that if we lift each of the paths of length at least 2 in $\mathcal{P}'$, then we obtain a graph G' such that for every $u, v \in V(H)$, we have that $\text{mult}_{G'}(\phi(u)\phi(v)) = \text{mult}_H(uv)$. Hence ϕ is an isomorphism from H to $G'[S]$. This shows that there exists a sequence of lifts $\mathcal{L}$ such that $G \vee \mathcal{L}$ contains H as an induced subgraph, implying that G contains H as an induced immersion. $\square$

We are now ready to state the main theorem of this section.

Theorem 2. *For every fixed multigraph H, there is a polynomial-time algorithm that decides for any graph G whether G contains H as an induced immersion.*

Proof. Let H be a fixed multigraph, and let G be a graph. Deciding whether G contains H as an induced immersion is equivalent to deciding whether G has an H-model due to Observation 2. We describe an

algorithm with running time $h(|V(H)| + |E(H)|) \cdot |V(G)|^{|V(H)|^2+3}$ that finds an H-model $(S, \mathcal{L}, \phi)$ of G, or decides that such a H-model does not exist. Note that this is a polynomial-time algorithm since H is a fixed multigraph.

Suppose G has an H-model $(S, \mathcal{L}, \phi)$. Then there exists a sequence of lifts $\mathcal{L}^*$ that satisfies conditions (i)–(iv) of Lemma 4. Let us determine an upper bound on the number of sequences of lifts $\mathcal{L}$ that satisfy conditions (i) and (ii). First note that the number of possible triples $u, v, w \in V(G)$ such that $\{uv, vw\}$ is a lift satisfying $v \in S$ and $\{u, w\} \cap S \neq \emptyset$ is at most $|V(H)|^2 \cdot |V(G)|$. Consequently, the number of sequences of at most $|E(G[S])| \leq |V(H)|^2$ such lifts is at most $(|V(H)|^2 \cdot |V(G)|)^{|V(H)|^2}$. Hence, there are at most $(|V(H)|^2 \cdot |V(G)|)^{|V(H)|^2}$ sequences of lifts that satisfy conditions (i) and (ii) of Lemma 4, and all these sequences can easily be generated in time $(|V(H)|^2 \cdot |V(G)|)^{|V(H)|^2}$.

For all possible subsets $S \subseteq V(G)$ of size $|V(H)|$ and all possible bijections ϕ from $V(H)$ to S, our algorithm acts as follows. For all possible sequences of lifts $\mathcal{L}$ that satisfy conditions (i) and (ii) of Lemma 4, the algorithm checks whether $\mathcal{L}$ satisfies (iii). If $\mathcal{L}$ does not satisfy (iii), then $\mathcal{L}$ is not a valid candidate for $\mathcal{L}^*$, and we can safely discard it. If $\mathcal{L}$ satisfies condition (iii), then we determine whether $\mathcal{L}$ satisfies (iv) as follows. We use the disjoint paths algorithm of Robertson and Seymour [16] in order to decide whether conditions (i) and (ii) of Lemma 5 hold for the graphs $G \vee \mathcal{L}$ and H. If so, then Lemma 5 guarantees that $G \vee \mathcal{L}$ contains H as an induced immersion, i.e., $\mathcal{L}$ satisfies condition (iv) of Lemma 4. This implies that G contains H as an induced immersion, so the algorithm outputs "yes". If $\mathcal{L}$ does not satisfy condition (iv), we proceed to the next sequence $\mathcal{L}$. If none of choices of S, ϕ and $\mathcal{L}$ yields a "yes"-answer, then we know from Lemma 4 that G does not have any H-model, and the algorithm outputs "no".

It remains to analyze the running time of the algorithm. There are at most $|V(G)|^{|V(H)|}$ subsets $S \subseteq V(G)$ of size $|V(H)|$, and for each of these sets S, there are $|V(H)|!$ bijections from $V(H)$ to S. As we saw earlier, there are at most $(|V(H)|^2 \cdot |V(G)|)^{|V(H)|^2}$ different sequences of lifts $\mathcal{L}$ that satisfy conditions (i) and (ii) of Lemma 4. This means the algorithm considers at most $f(|V(H)|) \cdot |V(G)|^{|V(H)|^2}$ combinations of S, ϕ and $\mathcal{L}$. For each of these combinations, testing whether condition (iii) holds can be done in time $O(|V(H)|^2(|E(H)|+1))$, while it takes $g(|E(H)|) \cdot |V(G)|^3$ time [16] to test whether condition (iv) holds, as we ask for edge-disjoint paths between at most $|E(H)|$ pairs of terminals. This yields an overall running time of $h(|V(H)| + |E(H)|) \cdot |V(G)|^{|V(H)|^2+3}$. □

4 Excluding a fixed multigraph as an induced immersion

In this section, we show that every multigraph whose treewidth is large enough contains every multigraph of maximum degree at most 2 as an induced immersion. We first show that every multigraph H is contained as an induced immersion in any multigraph G that contains a sufficiently large clique.

Lemma 6. *Let G and H be two multigraphs. If $K_{2(|V(H)|+|E(H)|)}$ is a subgraph of G, then G contains H as an induced immersion.*

Proof. Suppose G contains a clique X of size $2(|V(H)| + |E(H)|)$, and let $K = G[X]$. Let S be a set of $|V(H)|$ vertices of K, and let ϕ be an arbitrary bijection from $V(H)$ to S. We will apply lifts in order to make ϕ an isomorphism from $V(H)$ to S, i.e., we apply lifts to obtain a graph K^* such that $\mathrm{mult}_{K^*}(\phi(u)\phi(v)) = \mathrm{mult}_H(uv)$ for every $u, v \in V(H)$. For each pair of vertices $u, v \in V(H)$, we consider the multiplicity of uv in H and of $\phi(u)\phi(v)$ in K.

If $\mathrm{mult}_H(uv) = \mathrm{mult}_K(\phi(u)\phi(v))$, then we leave the set of edges between $\phi(u)$ and $\phi(v)$ unchanged.

For every $u, v \in V(H)$ with $\mathrm{mult}_H(uv) < \mathrm{mult}_K(\phi(u)\phi(v))$, we need to decrease the multiplicity of $\phi(u)\phi(v)$. To achieve this, we proceed in two steps. First, for every triple of (not necessarily distinct) vertices $\phi(u), \phi(v), \phi(w) \in S$, we lift $\{\phi(u)\phi(v), \phi(v)\phi(w)\}$ whenever $\mathrm{mult}_H(uv) < \mathrm{mult}_K(\phi(u)\phi(v))$ and $\mathrm{mult}_H(vw) < \mathrm{mult}_K(\phi(v)\phi(w))$, and we redefine $K = K \vee \{\phi(u)\phi(v), \phi(v)\phi(w)\}$. We apply this process as long as such triples $\phi(u), \phi(v), \phi(w)$ exist. Observe that this process terminates, as every lift decreases the number of edges in $K[S]$ by exactly 1. At the end of this process, for every vertex $u \in V(H)$, there is a most one vertex $v \in V(H)$ such that $\mathrm{mult}_H(uv) < \mathrm{mult}_K(\phi(u)\phi(v))$, and in that case we have $\mathrm{mult}_K(\phi(u)\phi(v)) = \mathrm{mult}_H(uv) + 1$. Hence there are at most $|V(H)|/2$ edges left to remove in $K[S]$. We now start the second step of our process. For every $u, v \in V(H)$ with $\mathrm{mult}_H(uv) < \mathrm{mult}_K(\phi(u)\phi(v))$, we define a set $Q_{uv} = \{w\}$, where w is an arbitrary vertex from $V(K) \setminus S$, in such a way that all the sets Q_{uv}, $u, v \in V(H)$, are pairwise disjoint. Note that these sets Q_{uv} exist, as the union Q of all sets Q_{uv} contains at most $|V(H)|/2$ vertices, while $V(K) \setminus S$ contains $|V(H)| + 2|E(H)|$ vertices. Now, for every set $Q_{uv} = \{w\}$, applying the lift $\{\phi(u)\phi(v), \phi(v)w\}$ decreases the multiplicity of $\phi(u)\phi(v)$ by 1. Doing so for every pair $u, v \in V(H)$ for which Q_{uv} is defined leaves us with a graph K^- in which $\mathrm{mult}_{K^-}(\phi(u)\phi(v)) \leq \mathrm{mult}_H(uv)$ for every pair $u, v \in V(H)$.

For every $u, v \in V(H)$ with $\text{mult}_H(uv) > \text{mult}_{K^-}(\phi(u)\phi(v))$, we need to increase the multiplicity of $\phi(u)\phi(v)$. For every such pair of vertices $u, v \in V(H)$, we define a set P_{uv}, consisting of arbitrary vertices of $V(K^-) \setminus S$, such that the following three conditions hold:

- if $u \neq v$, then $|P_{uv}| = \text{mult}_H(uv) - \text{mult}_{K^-}(\phi(u)\phi(v))$;
- if $u = v$, then $|P_{uv}| = 2(\text{mult}_H(uv) - \text{mult}_{K^-}(\phi(u)\phi(v)))$;
- all the sets P_{uv}, $u, v \in V(H)$, are pairwise disjoint.

Note that these sets P_{uv} exists, as the union P of all these sets contains at most $2 \cdot |E(H)|$ vertices, while $V(K^-) \setminus (S \cup Q)$ contains at least $2|E(H)| + |V(H)|/2$ vertices. Now, for every pair of vertices $u, v \in V(H)$ with $\text{mult}_H(uv) > \text{mult}_{K^-}(\phi(u)\phi(v))$, if $u \neq v$, then we apply the lift $\{\phi(u)x, x\phi(v)\}$ for each vertex $x \in P_{uv}$. Similarly, if $u = v$, then for each pair of distinct vertices $x, x' \in P_{uv}$, we first apply the lift $\{\phi(u)x, xx'\}$ and then use the newly created edge $\phi(u)x'$ to apply the lift $\{\phi(u)x', x'v\}$. After every step, we redefine K^- to be the obtained graph. It is clear that this process increases the multiplicity of $\phi(u)\phi(v)$ by exactly $\text{mult}_H(uv) - \text{mult}_{K^-}(\phi(u)\phi(v))$, and doing so for every pair $u, v \in V(H)$ for which P_{uv} is defined leaves us with a graph K^* in which $\text{mult}_{K^*}(\phi(u)\phi(v)) = \text{mult}_H(uv)$ for every $u, v \in V(H)$.

Since $K^*[S]$ is isomorphic to H and K^* can be obtained from K by lifting edges, Observation 1 implies that K contains H as an induced immersion. As K can be obtained from G by a sequence of vertex deletions, it follows that G contains H as an induced immersion. □

In an elementary wall W of height r, we define the i-th *row* of W to be the set of vertices $\{(j, i) \mid 0 \leq j \leq 2r + 1\}$. Similarly, the set $\{(j, i) \mid 0 \leq i \leq r\}$ is the j-th *column* of W. We write $P_i(j, k)$ to denote the unique path in W between (j, i) and (k, i) that contains only vertices of row i.

Lemma 7. *Let G be a multigraph, and let H be a multigraph of maximum degree at most 2. If G contains an elementary wall W as a subgraph such that $G[V(W)]$ contains an independent set of size $4|V(H)|(|V(H)| + 2)$, then G contains H as an induced immersion.*

Proof. Suppose G contains an elementary wall W as a subgraph such that $G[V(W)]$ contains an independent set S of size $4|V(H)|(|V(H)|+3)$. Since $4|V(H)|(|V(H)| + 2) \geq 2|V(H)|(2|V(H)| + 2) + 1$, at least $2|V(H)| + 3$ vertices of S lie on the same row of W, or at least $2|V(H)| + 1$ different

rows contain a vertex of S. We show that in both cases, G contains H as an induced immersion.

Let $H_1, \ldots, H_t$ be the connected components of H. We define $a_1 = 1$, and then recursively define $b_h = a_h + |V(H_h)| - 1$ and $a_h = b_{h-1} + 2$ for $1 \leq h \leq t$.

First suppose S contains at least $2|V(H)|+3$ vertices of row i. Let S^* be any subset of the vertices of S that lie on row i such that every vertex in S^* has two neighbors on row i and $|S^*| = 2|V(H)|+1$. We denote the vertices of S^* by (s_p, i), with $1 \leq p \leq 2|V(H)|+1$ and $s_p < s_{p+1}$ for every $1 \leq p \leq 2|V(H)|$. We now do as follows for each connected component H_h if $i \leq r-1$:

- if H_h is a path, we lift the path $P_i(s_p, s_{p+1})$ for every $a_h \leq p \leq b_h - 1$;
- if H_h is a cycle of length at least 2, we lift the path $P_i(s_p, s_{p+1})$, as well as the unique shortest path in W from s_{b_h} to s_{a_h} that is internally vertex-disjoint from $P_i(s_p, s_{p+1})$, for every $a_h \leq p \leq b_h - 1$;
- if H_h consists of a single vertex and a loop, we lift the unique 6-cycle in W that contains the edge $(s_{a_h}, i)(s_{a_h}+1, i)$ and three vertices of row $i+1$.

If $i = r$, then we replace $i+1$ by $i-1$ in the third case. Note that all the paths and cycles defined above exist due to the definitions of S^* and an elementary wall. Finally, we delete all the vertices from G that are not in S^*. The assumption that S^* is an independent set implies that the resulting graph is isomorphic to H, and therefore G contains H as an induced immersion.

Now suppose S contains vertices from at least $2|V(H)|+1$ different rows. Then there is a subset $S^* \subseteq S$ of size $2|V(H)|$ such that no two vertices of S^* lie on the same row of W and S^* does not contain any vertex of row r. We denote the vertices of S^* by (c_p, r_p), with $1 \leq p \leq 2|V(H)|$ and $r_p < r_{p+1}$ for every $1 \leq p \leq 2|V(H)| - 1$. We now do as follows for each connected component H_h:

- if H_h is a path, we lift a shortest path P_p in W from (c_p, r_p) to (c_{p+1}, r_{p+1}) for every $a_h \leq p \leq b_h - 1$, where we choose these paths in such a way that they are mutually internally vertex-disjoint.
- if H_h is a cycle, we lift a cycle in W that contains each of the vertices (c_p, r_p) with $a_h \leq p \leq b_h$ and contains only vertices of rows r_{a_h} to r_{b_h};

- if H_h consists of a single vertex and a loop, we lift the unique 6-cycle in W that contains (c_{a_h}, r_{a_h}) and three vertices of row $r_{a_h}+1$.

Note that all the paths and cycles defined above exist due to the definitions of S^* and an elementary wall. As in the previous case, deleting all the vertices of G that do not belong to S^* yields a graph isomorphic to H, which implies that G contains H as an induced immersion. □

We are now ready to prove the main result of this section.

Theorem 3. *For every multigraph H of maximum degree at most 2, there exists a constant c_H such that every multigraph with treewidth more than c_H contains H as an induced immersion.*

Proof. Let G and H be two multigraphs such that G does not contain H as an induced immersion. Let r be the largest integer such that G contains the elementary wall W_r as an immersion. Then there is a sequence $\mathcal{X}$ of vertex deletions, edge deletions and lifts such that applying $\mathcal{X}$ to G yields W_r. Let W' be the graph obtained from G by applying only the vertex deletions and lifts in $\mathcal{X}$. Then G contains W' as an induced immersion. Note that W' contains W_r as a spanning subgraph. Since G does not contain H as an induced immersion, Lemma 7 implies that W' does not have an independent set of size $4|V(H)|(|V(H)|+2)$. In addition, we know that W' does not contain a clique of size $2(|V(H)|+|E(H)|)$ as a subgraph, as otherwise G would contain H as an induced immersion as a result of Lemma 6. Ramsey's Theorem (cf. [4]) states that a graph that has neither a clique nor an independent set of size more than k must have at most 2^{2k-3} vertices. Hence, we know that W' has at most 2^{2k-3} vertices, where $k \leq \max\{4|V(H)|(|V(H)|+2), 2(|V(H)|+|E(H)|)\}$. Since H has maximum degree at most 2, we have $|E(H)| \leq 2|V(H)|$ and consequently $k \leq 4|V(H)|(|V(H)|+2)$. On the other hand, by the definition of an elementary wall of height r, we know that W' has exactly $2(r+1)^2-2$ vertices. Therefore we obtain that $2(r+1)^2-2 \leq 2^{2k-3}$, which implies $r \leq 2^k$. By the definition of r, G does not contain W_{r+1} as an immersion. Hence, by Theorem 1, the treewidth of G is at most $20^{10(r+2)^5} \leq 20^{10(2^k+2)^5}$. We conclude that every multigraph with treewidth more than $20^{10(2^{4|V(H)|(|V(H)|+2)}+2)^5}$ contains H as an induced immersion. □

5 Concluding remarks

It is not hard to show that every induced immersion of an elementary wall is a planar graph with maximum degree at most 3, and that an

elementary wall of height c has treewidth at least c. This implies that we cannot replace "maximum degree at most 2" in Theorem 3 by "maximum degree at most 4". A natural question is whether Theorem 3 holds for every planar multigraph H of maximum degree at most 3.

Our results exhibit some interesting relations between induced immersions and immersions. For example, Lemma 5 readily implies the following result.

Corollary 1. *Let G and H be two multigraphs. If there is a set $S \subseteq V(G)$ such that $G[S]$ is isomorphic to a spanning subgraph of H, then G contains H as an induced immersion with branch vertices S if and only if G contains H as an immersion with branch vertices S.*

We can also note the following corollary of Lemma 6.

Corollary 2. *Let G and H be two multigraphs. If G contains the graph $K_{2(|V(H)|+|E(H)|)}$ as an immersion, then G contains H as an induced immersion.*

Proof. Assume that G contains $K_{2(|V(H)|+|E(H)|)}$ as an immersion. Then the graph $K_{2(|V(H)|+|E(H)|)}$ can be obtained from G by a sequence $\mathcal{X}$ of lifts, vertex deletions and edge deletions. Let G' be the multigraph obtained from G by applying only the lifts and vertex deletions in $\mathcal{X}$. This multigraph G' contains $K_{2(|V(H)|+|E(H)|)}$ as a spanning subgraph. Then G', and hence G, contains H as an induced immersion as a result of Lemma 6. □

Corollary 2 implies that forbidding a graph as an induced immersion also forbids another graph as an immersion. In particular, the structure theorem for graphs excluding a clique of fixed size as an immersion [20] can also be applied to induced immersions, at the cost of larger constants. Moreover, every class of graphs closed under taking induced immersions has bounded degeneracy [2].

Two very interesting and challenging questions on induced immersions remain. In terms of parameterized complexity [6], Theorem 2 states that Induced Immersion is in XP when parameterized by the size of H, i.e., $|V(H)| + |E(H)|$. A natural question is whether the problem is fixed-parameter tractable when parameterized by the size of H. Very recently, Grohe et al. [13] established fixed-parameter tractability of the closely related Immersion problem, thereby resolving a longstanding open question by Downey and Fellows [5]. Interestingly, the next lemma shows that an FPT algorithm for Induced Immersion would immediately imply an

FPT algorithm for Immersion, rendering the problem of finding such an FPT algorithm a very challenging one.

Let P and Q be two parameterized problems. A *parameterized reduction* from P to Q is a function that maps an instance (x, k) of problem P to an instance (x', k') of problem Q in time $f(k) \cdot |x|^{O(1)}$ for some function f, where k is the parameter of x, such that (x, k) is a yes-instance of P if and only if (x', k') is a yes-instance of Q, and $k' \leq g(k)$ for some function g, where k' is the parameter of x'. If there is a parameterized reduction from P to Q and Q is FPT, then P is also FPT [6].

Lemma 8. *There exists a parameterized reduction from* Immersion *to* Induced Immersion *if both problems are parameterized by the size of* H*.*

Proof. Let G and H be two multigraphs. We construct a multigraph G^* from G by adding, for every vertex $v \in V(G)$, $|E(G)| + 2$ new vertices that are all made adjacent to v only. We refer to the newly added vertices as *red* vertices, and to the other vertices of G^* as *black* vertices. We write B_{G^*} to denote the set of black vertices in G^*. Any edge of G^* incident with a red vertex is called a red edge, and any other edge of G^* is black. We construct a multigraph H^* from H as follows: for every vertex v of degree 1, we add a new vertex that is made adjacent to v only, for every isolated vertex w, we add two new vertices that are made adjacent to w only. Black and red vertices and edges in H^* are defined in the same way as in G^*, and we denote the set of black vertices in H^* by B_{H^*}.

We prove that G contains H as an immersion if and only if G^* contains H^* as an induced immersion.

First suppose G contains H as an immersion. Then there exists a sequence of edge deletions, vertex deletions and lifts, denoted by $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$, respectively, such that applying $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ to G yields a graph that is isomorphic to H. We now describe a sequence of lifts $\mathcal{L}$ such that the graph $G^* \vee \mathcal{L}$ contains H^* as an induced subgraph. First, apply all the lifts in $\mathcal{X}_3$ to G^*. In the obtained graph $G^* \vee \mathcal{X}_3$, we simulate the deletion of every edge uv in $\mathcal{X}_1$ by lifting uv in $G^* \vee \mathcal{X}_3$ with any pendant red edge incident with either u or v; since every vertex in B_{G^*} is incident with $|E(G)| + 2$ red edges in G^*, there is always an available red edge incident with either u or v. Let G' be the graph obtained from G^* this way. Observe that $G'[B_{G^*}]$ contains H as an induced subgraph. Moreover, by the construction of G^*, every black vertex in G' has at least two pendant red edges incident to it. Hence, it is clear that G' contains H^* as an induced subgraph. Since G' is obtained from G^* by a sequence of lifts, Observation 1 implies that G^* contains H^* as an induced immersion.

For the reverse direction, suppose G^* contains H^* as an induced immersion. Then, by Observation 1, there exists a sequence of lifts $\mathcal{L}$ and a set $S \subseteq V(G)$ such that $G'[S]$ is isomorphic to H^*, where $G' = G^* \vee \mathcal{L}$. Note that every red vertex has degree 1 in G^* and will also have degree 1 in G', as we only apply lifts when transforming G^* into G'. Hence $H^*[B_{H^*}]$ is an induced subgraph of $G'[B_{G^*}]$, as every vertex in B_{H^*} has degree at least 2 in H^*. In order to show that G contains H as an immersion, we describe a sequence of lifts and edge deletions whose application to G yields a graph that contains H as an induced subgraph. For every lift in $\mathcal{L}$ consisting of two black edges of G^*, we lift the same edges in G; for every lift in $\mathcal{L}$ consisting of a black edge and a red edge of G^*, we delete the lifted black edge in G; and for every lift in $\mathcal{L}$ consisting of two red edges of G^*, we leave G unaltered. Let G'' denote the resulting graph. Since $H^*[B_{H^*}]$ is an induced subgraph of $G'[B_{G^*}]$, we have that G'' contains H as an induced subgraph. We conclude that G contains H as an immersion.

In order to see that the transformation described above is a parameterized reduction, it suffices to observe that $|V(H^*)| + |E(H^*)| \leq 3|V(H)|+2|E(H)|$, and that the transformation can clearly be performed in $f(|V(H)| + |E(H)|) \cdot (|V(G)| + |E(G)|)^{O(1)}$ time, since $|V(G^*)| = |V(G)| \cdot (|E(G)| + 2)$ and $|E(G^*)| = |E(G)| + |V(G)| \cdot (|E(G)| + 2)$. $\square$

Another celebrated result on immersions, due to Robertson and Seymour [18], states that all graphs are well-quasi ordered with respect to the immersion relation. Does the same hold for induced immersions? If so, then proving such a statement seems to be a formidable task, as it would imply the aforementioned result of Robertson and Seymour. An easier task, recently proposed by Fellows, Hermelin and Rosamond [7], would be to identify interesting classes of graphs that are well-quasi ordered under the induced immersion relation.

References

[1] Damaschke, P.: Induced Subgraph Isomorphism for Cographs in NP-Complete. In: WG 1990, LNCS 484, pp. 72–78 (1990)

[2] DeVos, M., Dvorak, Z., Fox, J., McDonald, J., Mohar, B. and Scheide, D.: Minimum degree condition forcing complete graph immersion. Submitted for publication.

[3] DeVos, M., Kawarabayashi, K., Mohar, B., Okamura, H.: Immersing small complete graphs. Ars Math. Contemp. 3:139–146 (2010)

[4] Diestel, R.: Graph Theory. Springer-Verlag, Electronic Edition (2005)

[5] Downey, R.G., Fellows, M.R.: Fixed-parameter intractability. In: Structure in Complexity Theory Conference, pp. 36–49 (1992)

[6] Downey, R.G., Fellows, R.: Parameterized Complexity. Monographs in Computer Science. Springer (1999)

[7] Fellows, M.R., Hermelin, D., Rosamond, F.A.: Well quasi orders in subclasses of bounded treewidth graphs and their algorithmic applications. Algorithmica 64:3–18 (2012)

[8] Fellows, M.R., Kratochvíl, J., Middendorf, M., Pfeiffer, F.: The complexity of induced minors and related problems. Algorithmica 13:266–282 (1995)

[9] Fellows, M.R., Langston, M.A.: On well-partial-order theory and its application to combinatorial problems of VLSI design. SIAM J. Discrete Math. 5(1):117–126 (1992)

[10] Ferrara, M.,Gould, R.,Tansey, G.,Whalen, T.: On H-immersions. J. Graph Theory 57, 245–254 (2008)

[11] Garey, M.R., Johnson, D.S., Tarjan, R.E.: The planar Hamiltonian circuit problem is NP-complete. SIAM J. Computing 5(4):704–714 (1976)

[12] Giannopoulou, A., Kamiński, M., Thilikos, D. M.: Forbidding Kuratowski graphs as immersions. In preparation.

[13] Grohe, M., Kawarabayashi, K., Marx, D., Wollan, P.: Finding topological sugraphs is fixed-parameter tractable. In: STOC 2011, pp. 479–488. ACM (2011)

[14] Grohe, M., Marx, D.: Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In: STOC 2012, pp. 173–192. ACM (2012)

[15] Lévêque, B., Lin, D. Y., Maffray, F., Trotignon, N.: Detecting induced subgraphs. Electronic Notes in Discrete Mathematics 29:207–211 (2007)

[16] Robertson, N., Seymour, P.D.: Graph minors XIII: The disjoint paths problem. J. Comb. Theory, Ser. B 63(1):65–110 (1995)

[17] Robertson, N., Seymour, P.D.: Graph minors XX: Wagner's conjecture. J. Comb. Theory, Ser. B 92(2):325–357 (2004)

[18] Robertson, N., Seymour, P.D.: Graph Minors XXIII: Nash-Williams' Immersion Conjecture. J. Comb. Theory, Ser. B 100(2):181–205 (2010)

[19] Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. J. Comb. Theory, Ser. B 62:323–348 (1994)

[20] Seymour, P.D., Wollan, P.: The structure of graphs not admitting a fixed immersion. Manuscript.

Chapter 9

Structure of Wheel-immersion-free Graphs

Structure of W_4-immersion free graphs

Rémy Belmonte[1] Archontia C. Giannopoulou[1]
Daniel Lokshtanov[1] Dimitrios M. Thilikos[2]

[1] Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway
{Remy.Belmonte,Archontia.Giannopoulou,daniello}@ii.uib.no

[2] CNRS, LIRMM and
Department of Mathematics, National & Kapodistrian University of Athens,
Panepistimioupolis, GR-15784, Athens, Greece
sedthilk@thilikos.info

Abstract

We study the structure of graphs that do not contain the wheel on 5 vertices W_4 as an immersion, and show that these graphs can be constructed via 1, 2, and 3-edge-sums from subcubic graphs and graphs of bounded treewidth.

Keywords: Immersion Relation, Wheel, Treewidth, Edge-sums, Structural Theorems.

1 Introduction

A recurrent theme in structural graph theory is the study of specific properties that arise in graphs when excluding a fixed pattern. The notion of appearing as a pattern gives rise to various graph containment relations. Maybe the most famous example is the minor relation that has been widely studied, in particular since the fundamental results of Kuratowski and Wagner who proved that planar graphs are exactly those graphs that contain neither K_5 nor $K_{3,3}$ as a (topological) minor. A graph G contains a graph H as a topological minor if H can be obtained from G by a sequence of vertex deletions, edge deletions and replacing internally vertex-disjoint paths by single edges. Wagner also described the structure of the graphs that exclude K_5 as a minor: he proved that K_5-minor-free

graphs can be constructed by "gluing" together (using so-called clique-sums) planar graphs and a specific graph on 8 vertices, called Wagner's graph.

Wagner's theorem was later extended in the seminal Graph Minor series of papers by Robertson and Seymour (see e.g. [23]), which culminated with the proof of Wagner's conjecture, i.e., that graphs are well-quasi-ordered under minors [24], and ended with the proof of Nash-Williams' immersion conjecture, i.e., that the graphs are also well-quasi-ordered under immersions [25]. Other major results in graph minor theory include the (Strong) Structure Theorem [23], the Weak Structure Theorem [22], the Excluded Grid Theorem [17, 21, 26], as well as numerous others, e.g., [5, 15, 28]. Moreover, the structural results of graph minor theory have deep algorithmic implications, one of the most significant examples being the existence of cubic time algorithms for the k-Disjoint Paths and H-Minor Containment problems [22]. For more applications see, e.g., [1, 3, 6, 16, 18].

However, while the structure of graphs that exclude a fixed graph H as a minor has been extensively studied, the structure of graphs excluding a fixed graph H as a topological minor or as an immersion has not received as much attention. While a general structure theorem for topological minor free graphs was very recently provided by Grohe and Marx [14], finding an exact characterization of the graphs that exclude K_5 as a topological minor remains a notorious open problem. Recently, Wollan gave a structure theorem for graphs excluding complete graphs as immersions [29]. A graph G contains a graph H as a immersion if H can be obtained from G by a sequence of vertex deletions, edge deletions and replacing edge-disjoint paths by single edges. Observe that if a graph G contains a graph H as a topological minor, then G also contains H as an immersion, as vertex-disjoint paths are also edge-disjoint. In 2011, DeVos et al. [7] proved that if the minimum degree of a graph G is at least $200t$ then G contains the complete graph on t vertices as an immersion. In [10] Ferrara et al. provided a lower bound on the minimum degree of any graph G in order to ensure that a given graph H is contained in G as an immersion.

A common drawback of such general results is that they do not provide sharp structural characterizations for concrete instantiations of the excluded graph H. In the particular case of immersion, such structural results are only known when excluding both K_5 and $K_{3,3}$ as immersions [11]. In this paper, we prove a structural characterization of the graphs that exclude W_4 as an immersion and show that they can be constructed from graphs that are either subcubic or have treewidth bounded by a constant.

We denote by W_4 the wheel with 4 spokes, i.e., the graph obtained from a cycle on 4 vertices by adding a universal vertex. The structure of graphs that exclude W_4 as a topological minor has been studied by Farr [9]. He proved that these graphs can be constructed via clique-sums of order at most 3 from graphs that are either subcubic or have diameter at most 3. However, this characterization only applies to simple graphs. In our study we exclude W_4 as an immersion while allowing multiple edges.

As with the minor relation, many algorithmic results have also started appearing in terms of immersions. In [13], Grohe et al. gave a cubic time algorithm that decides whether a fixed graph H immerses in any input graph G. This algorithm, combined with the well-quasi-ordering of immersions [25], implies that the membership of a graph in any graph class that is closed under taking immersions can be decided in cubic time. However, the construction of such an algorithm requires the ad-hoc knowledge of the finite set of excluded immersions that characterizes this graph class (which is called obstruction set). While no general way to compute an obstruction set is known, in [12], Giannopoulou et al. proved that the obstruction set of an immersion-closed graph class can be computed when an upper bound on the treewidth of the obstructions and a description of the graph class in Monadic Second Order Logic are given. Another example of explicit construction of immersion obstruction sets is given by Belmonte et al. [2], where the set of immersion obstructions is given for graphs of carving-width 3.

Our paper is organized as follows: in Section 2, we give necessary definitions and previous results. In Section 3, we show that containment of W_4 as an immersion is preserved under 1, 2 and 3-edge-sums. Then, in Section 4, we provide our main result, i.e., a decomposition theorem for graphs excluding W_4 as an immersion. Finally, we conclude with remarks and open problems.

2 Preliminaries

For undefined terminology and notation, we refer to the textbook of Diestel [8]. For every integer n, we let $[n] = \{1, 2, \ldots, n\}$. All graphs we consider are finite, undirected, and without self-loops but may have multiple edges. Given a graph G we denote by $V(G)$ and $E(G)$ its *vertex* and *edge set* respectively. Given a set $F \subseteq E(G)$ (resp. $S \subseteq V(G)$), we denote by $G \setminus F$ (resp. $G \setminus S$) the graph obtained from G if we remove the edges in F (resp. the vertices in S along with their incident edges). We denote by $\mathcal{C}(G)$ the set of the *connected components* of G. Given two vertices $u, v \in V(G)$, we also use the notation $G - v = G \setminus \{v\}$ and the

notation uv for the edge $\{u, v\}$. The *neighborhood* of a vertex $v \in V(G)$, denoted by $N_G(v)$, is the set of vertices in G that are adjacent to v. We denote by $E_G(v)$ the set of the edges of G that are incident with v. The *degree* of a vertex $v \in V(G)$, denoted by $\deg_G(v)$, is the number of edges that are incident with it, that is, $\deg_G(v) = |E_G(v)|$. Notice that, as we are working with multigraphs, $|N_G(v)| \leq \deg_G(v)$. The degree of a set S, denoted by $\partial(S)$, is the number of edges between S and $V(G) \setminus S$, that is $|\{uv \in E(G) \mid u \in S \wedge v \notin S\}|$. Given two vertices u and v with $u \in N(v)$ we say that u is an i-neighbor of v if $E(G)$ contains exactly i copies of the edge $\{u, v\}$. Let P be a path and $v, u \in V(P)$. We denote by $P[v, u]$ the subpath of P with endpoints v and u. The *maximum degree* of a graph G, denoted by $\Delta(H)$ is the maximum of the degrees of the vertices of G, that is, $\Delta(G) = \max_{v \in V(G)} \deg_G(v)$.

We denote by W_{k-1} the *wheel* on k vertices, that is, the graph obtained from the cycle of length $k-1$ after adding a new vertex and making it adjacent to all of its vertices. We call the new vertex *center* of the wheel.

Definition 1. *An immersion of H in G is a function α with domain $V(H) \cup E(H)$, such that:*

- *$\alpha(v) \in V(G)$ for all $v \in V(H)$, and $\alpha(u) \neq \alpha(v)$ for all distinct $u, v \in V(H)$;*
- *for each edge e of H, $\alpha(e)$ is a path of G with ends $\alpha(u), \alpha(v)$;*
- *for all distinct $e, f \in E(H), E(\alpha(e) \cap \alpha(f)) = \emptyset$.*

We call the image of every such function α in G *model* of the graph H in G and the vertices of the set $\alpha(V(H))$ *branch* vertices of α.

An *edge cut* in a graph G is a non-empty set F of edges that belong to the same connected component of G and such that $G \setminus F$ has more connected components than G. If $G \setminus F$ has one more connected component than G and no proper subset of F is an edge cut of G, then we say that F is a *minimal* edge cut. Given a connected set S such that $G \setminus S$ is also connected, we denote by $(S, G \setminus S)$ the cut between S and $G \setminus S$. Let F be an edge cut of a graph G and let G be the connected component of G containing the edges of F. We say that F is an *internal* edge cut if it is minimal and both connected components of $G \setminus F$ contain at least 2 vertices. An edge cut is also called *i-edge cut* if it has order at most i.

Definition 2. *Let G, G_1, and G_2 be graphs. Let $t \geq 1$ be a positive integer. The graph G is a t-edge-sum of G_1 and G_2 if the following holds. There exist vertices $v_i \in V(G_i)$ such that $|E_{G_i}(v_i)| = t$ for $i \in [2]$*

and a bijection $\pi : E_{G_1}(v_1) \to E_{G_2}(v_2)$ *such that* G *is obtained from* $(G_1 - v_1) \cup (G_2 - v_2)$ *by adding an edge from* $x \in V(G_1) - v_1$ *to* $y \in V(G_2) - v_2$ *for every pair of edges* e_1 *and* e_2 *such that* $e_1 = xv_1$, $e_2 = yv_2$, *and* $v_2 = \pi(e_1)$. *We say that the edge-sum is internal if both* G_1 *and* G_2 *contain at least 2 vertices and denote the internal* t*-edge-sum of* G_1 *and* G_2 *by* $G_1 \hat{\oplus}_t G_2$.

We say that an t-edge-sum is *minimal* if there is no t'-edge-sum such that $t' < t$. Note that if G is the minimal t-edge-sum of graphs G_1 and G_2 for some $t \geq 0$, then the set of edges $\{\{u, v\} \in E(G) \mid u \in V(G_1), v \in V(G_2)\}$ forms a minimal edge cut of G of order t. Moreover, we may always assume that t-edge-sums are minimal, and hence that $G_1 \setminus v_1$ and $G_2 \setminus v_2$ are connected.

Let r be a positive integer. The (r, r)*-grid* is the graph with vertex set $\{(i, j) \mid i, j \in [r]\}$ and edge set $\{\{(i, j), (i', j')\} \mid |i - i'| + |j - j'| = 1\}$. The *(elementary) wall* of height r is the graph W_r with vertex set $V(W_r) = \{(i, j) \mid i \in [r + 1], j \in [2r + 2]\}$ in which we make two vertices (i, j) and (i', j') adjacent if and only if either $i = i'$ and $j' \in \{j - 1, j + 1\}$ or $j' = j$ and $i' = i + (-1)^{i+j}$, and then remove all vertices of degree 1; see Figure 1 for some examples. The vertices of this vertex set are called *original* vertices of the wall. A *subdivided wall* of height r is the graph obtained from W_r after replacing some of its edges by internally vertex-disjoint paths.

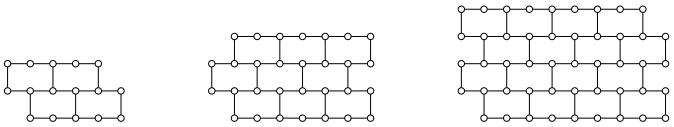


Figure 1: Elementary walls of height 2, 3, and 4.

Let r be a positive integer and notice that the wall of height r is contained in the $((2r + 2) \times (2r + 2))$-grid as a subgraph. This implies that any graph containing the $((2r + 2) \times (2r + 2))$-grid as a minor also contains the wall of height r as a minor. Furthermore, from a folklore result, for any simple graph H such that $\Delta(H) \leq 3$ it holds that H is a minor of a graph G if and only if H is a topological minor of G.

Theorem 1. *[19] Let* G *and* H *be two graphs, where* H *is connected and simple, not a tree, and has* h *vertices. Let also* g *be a positive integer. If* G *has treewidth greater than* $3(8h(h-2)(2g+h)(2g+1))^{|E(H)|-|V(H)|} + \frac{3h}{2}$ *then* G *contains either the* $(g \times g)$*-grid or* H *as a minor.*

Theorem 1, in the case where $g = 2r + 2$ and H is the wall of height r, can be restated as the well known fact that large treewidth ensures the existence of a large wall as a topological minor:

Theorem 2. *[19] Let G be a graph and $r \geq 2$ be an integer. If the treewidth of G is greater than $2^{18r^2 \log r}$ then G contains the wall of height r as a topological minor.*

3 Invariance of W_4 containment under small edge-sums

In this section, we show that immersion of W_4 is completely preserved under edge-sums of order at most 3, i.e., that W_4 immerses in a graph G if and only if it immerses in at least one of the graphs obtained by decomposing G along edge-sums. Theorem 3 will be necessary in Section 4 to ensure that our decomposition does not change whether the graphs considered contain W_4 as an immersion or not. We first prove the following general lemma.

Lemma 1. *If G, G_1, and G_2 are graphs such that $G = G_1 \hat{\oplus}_t G_2$, $t \in [3]$, then both G_1 and G_2 are immersed in G.*

Proof. Notice that it is enough to prove that G_1 is an immersion of G. Let v_1 and v_2 denote the unique vertex of $V(G_1) \setminus V(G)$ and $V(G_2) \setminus V(G)$ respectively. In the case where $G = G_1 \hat{\oplus}_1 G_2$, let u_i be the unique neighbor of v_i in G_i, $i \in [2]$. Then the function $\{(v, v) \mid v \in V(G_1 - v_1)\} \cup \{(v_1, u_2)\}$ is an isomorphism from G_1 to the graph $G \setminus V(G_2 - u_2)$ (by the definition of the edge-sum $u_2u_1 \in E(G)$) which is a subgraph of G. Therefore, $G_1 \subseteq G$ and thus G_1 also immerses in G.

We now assume that $G = G_1 \hat{\oplus}_t G_2$, $t = 2, 3$. Let e_j, $j \in [|E_{G_1}(v_1)|]$, be the edges of $E_{G_1}(v_1)$ and let u_j be the (not necessarily distinct) endpoints of the edges e_j, $j \in [|E_{G_1}(v_1)|]$, in $G_1 - v_1$. Notice that in both cases, in order to obtain G_1 as an immersion of G, it is enough to find a vertex u in $V(G) \setminus V(G_1)$ and for each edge e_j of $E_{G_1}(v_1)$ find a path P_j from u to e_j in $E(G) \setminus E(G_1)$ such that these paths are edge-disjoint. In what follows we find such vertex and paths. We distinguish the following cases.

Case 1. $N_{G_2}(v_2) = \{y\}$. Then, by the definition of the edge-sum, G contains the edges ye_j^1, $j \in [|E_{G_1}(v_1)|]$. Notice that neither the vertex y belongs to $V(G_1)$ nor the edges yu_j^1, $j \in [|E_{G_1}(v_1)|]$, belong to $E(G_1)$ and therefore the claim holds for $u = y$.

Case 2. $N_{G_2}(v_2) = \{x, y\}$. First notice that in the case where $G = G_1 \hat{\oplus}_3 G_2$ one of the x, y, say x, is a 2-neighbor of v_2. As the edge-sum is internal, the set $E = E(G) \setminus (E(G_1) \cup E(G_2))$ of edges created after the edge-sum is a minimal separator of G. Without loss of generality let yu_1^1, xu_2^1, and (in the case where $G = G_1 \hat{\oplus}_3 G_2$) xu_3^1 be its edges. By the minimality of the separator E, $G_2 - v_2$ is connected. Therefore there exists a (x, y)-path P in $G_2 - v_2$. Observe that the path $P \cup \{yu_1^1\}$, the path consisting only of the edge xu_2^1 and (in the case where $G = G_1 \hat{\oplus}_3 G_2$) the path consisting only of the edge xu_3^1 are edge-disjoint paths who do not have any edge from $E(G_1)$ and share x as a common endpoint. Then the claim holds for $u = x$.

Case 3. $N_{G_2}(v_2) = \{x, y, z\}$. In this case, it holds that $G = G_1 \hat{\oplus}_3 G_2$. As above, consider the set $E = E(G) \setminus (E(G_1) \cup E(G_2))$ of the edges created by the edge-sum and without loss of generality, let $E = \{xu_1, yu_2, zu_3\}$. Since E is a minimal separator, the graph $G_2 - v_2$ is connected. Therefore, there are a (x, y)-path P and a (y, z)-path Q in $G_2 - v_2$. Let z' be the vertex in $V(P) \cap V(Q)$ such that $V(Q[z, z']) \cap V(P) = \{z'\}$ and consider the paths $Q[z, z']$, $P[x, z']$, and $P[z', y]$ (in the case where $z' = y$ the path $P[z', y]$ is the graph consisting of only one vertex). Observe that these graphs are edge-disjoint. Therefore the paths $P[x, z'] \cup \{xu_1\}$, $P[y, z'] \cup \{yu_2\}$, and $Q[z, z'] \cup \{zu_3\}$ are edge-disjoint, do not contain any edge from $E(G_1)$, and share the vertex z' as an endpoint. Thus, the claim holds for $u = z'$. It then follows that G_1 is an immersion of G and this completes the proof of the lemma. □

Theorem 3. *Let G, G_1, and G_2 be graphs such that $G = G_1 \hat{\oplus}_t G_2$, with $t \in [3]$. Then, G contains W_4 as an immersion if and only if G_1 or G_2 does as well.*

Proof. If G_1 or G_2 contains W_4 as an immersion, then G does as well due to Lemma 1. It remains to prove the converse direction.

Let α be an immersion of W_4 in G. We first prove that either $|\alpha(V(W_4)) \cap (V(G_1) - v_1)| \geq 4$, or $|\alpha(V(W_4)) \cap (V(G_2) - v_2)| \geq 4$. Indeed, this is due to the fact that any cut $(S, G \setminus S)$ of W_4 with $|S| = 3$ has order at least 4, whereas the cut $F = E(G) \setminus (E(G_1) \cup E(G_2))$ in G between $V(G_1) - v_1$ and $V(G_2) - v_2$ has order at most 3. Moreover, the same argument implies that the image of the center of W_4, that is, the unique vertex of degree 4 of W_4, say x_0, belongs to the connected component of $G - F$ that contains at least 4 of the branch vertices of the immersion α. Let us assume without loss of generality that $x_0 \in V(G_1) - v_1$.

Assume first that $\alpha(V(W_4)) \cap (V(G_1) - v_1) = 5$. If for every edge e of W_4 it holds that $\alpha(e) \cap V(G_2 - v_2) = \emptyset$, then clearly α is an immersion of W_4 in $G_1 - v_1$, and therefore in G_1. Moreover, it is easy to observe that there cannot be two distinct edges e, e' of W_4 whose image path in G contains vertices of $G_2 - v_2$, since each such path must contain at least 2 edges of F, and $|F| \leq 3$. Hence we may assume that there exists a unique edge e with $\alpha(e) \cap V(G_2 - v_2) \neq \emptyset$. Note that $\alpha(e)$ must intersect the cut F in an even number of edges, since otherwise the path would end in $G_2 - v_2$, contradicting our assumption that all branch vertices of α lie in $G_1 - v_1$. Let P be the maximum subpath of $\alpha(e)$ such that $E(P') \cap E(G_1 - v_1) = \emptyset$. Notice that the first and the last edge of such a path are edges of F. Let u_1 and u_2 be the endpoints of P. This implies that we may obtain an immersion α' of W_4 in G_1 by replacing in α the path P by the path $u_1v_1u_2$.

Now, we assume that $\alpha(V(W_4)) \cap (V(G_1) - v_1) = 4$, and denote by x the unique branch vertex of α lying in $V(G_2 - v_2)$. We claim that it is possible to create an immersion function α' of W_4 in G_1 by replacing the vertex x in α with v_1. To show this, we apply the following operations to G: let P_1, P_2, P_3 be the paths of α whose associated edges in W_4 are incident with $\alpha^{-1}(x_4)$, and let P_1', P_2', P_3' be the subpaths of P_1, P_2, and P_3 that do not contain edges of $G_1 - v_1$. The paths P_1', P_2', P_3' are easily observed to be edge-disjoint, and therefore we may lift the edges in each of these paths. We complete the construction by deleting the vertices in $V(G_2) - \{v_2, x_4\}$. The graph obtained from this construction is readily observed to be isomorphic to G_1 by mapping every vertex of $G_1 - v_1$ to itself, and v_1 to x. Therefore W_4 immerses in G_1. This concludes the proof of the theorem. □

4 Structure of graphs excluding W_4 as an immersion

In this section, we prove the main result of our paper, namely we provide a structure theorem for graphs that exclude W_4 as an immersion. We first provide a technical lemma that will be crucial for the proof of Theorem 5.

Lemma 2. *There exists a function f such that for every integer $r \geq 60000$ and every graph G that does not contain W_4 as an immersion, has no internal 3-edge cut, and has a vertex u with $d(u) \geq 4$, if $tw(G) \geq f(r)$, then there exist sets $Z = \{z_1, \ldots, z_r\}$, $S_1, \ldots, S_r$, and X, that satisfy the following properties:*

(i) $z_i \in S_i, \forall i \in \{1, \ldots, r\}$;

(ii) $z_i \notin S_j, \forall i \neq j \in \{1, \ldots, r\}$;

(iii) $u \in \bigcap_{i \in \{1,\ldots,r\}} S_i$;

(iv) $\partial(S_i) \leq 6$;

(v) $G[S_i]$ *is connected,* $\forall i \in \{1, \ldots, r\}$;

(vi) $X \cap S_i = \emptyset, \forall i \in \{1, \ldots, r\}$;

(vii) *For every* $Z' \subseteq Z$ *such that* $|Z'| \geq 7$*, there is a 7-flow from* Z' *to* X;

Proof. Assume that G has treewidth at least $2^{18(6r)^2 \log(6r)}$, Then, from Theorem 2, $G - u$ contains an elementary wall of height $6r$ as a topological minor. We define the cycles $C_1, \ldots, C_{6r}$ as the ones formed by vertices $w_{5+20p,3+2q}$ to $w_{11+20p,3+2q}$ and $w_{11+20p,4+2q}$ to $w_{5+20p,4+2q}$, for every $p, q \in \{0, \ldots, \lceil\sqrt{6r}\rceil - 1\}$. Observe that $C_1, \ldots, C_{6r}$ is a set of vertex disjoint cycles containing at least 14 vertices in $G - u$. For every $i \in [6r]$, we denote by G_{C_i} the graph obtained from G by removing the edges of C_i and adding a vertex v_i adjacent exactly to the vertices of C_i. Since W_4 does not immerse in G, there exists an edge cut F_i of order at most 3 that separates u and v_i, and since both u and v_i have degree at least 4, this edge cut is internal. We now define the set T_i, for every $i \in [6r]$, as the set of vertices that lie in the same connected component of $G_{C_i} - F_i$ as u.

Claim 1. *For every* $i \in \{1, \ldots, 6r\}$, $1 \leq |T_i \cap V(C_i)| \leq 3$.

Proof of Claim 1. The fact that $|T_i \cap V(C_i)| \geq 1$ follows from the observation that if $T_i \cap V(C_i) = \emptyset$, then the cut F_i is not only a cut in G_{C_i}, but also in G, which contradicts the assumption that G is internally 4 edge-connected. On the other hand, observe that for every vertex $w \in V(C_i) \cap T_i$, the edge vw must belong to the cut F_i. Therefore no more than 3 vertices of C_i may lie in T_i, which concludes the proof of the claim. ◇

We now define the set $Z = \{z_1, \ldots, z_{6r}\}$: for every $i \in [6r]$, we choose arbitrarily one vertex of $T_i \cap V(C_i)$ to be the vertex z_i. The existence of the vertices z_i follows from Claim 1. Observe that, by construction of Z, it holds that $z_i \in T_i, \forall i \in [6r]$, i.e., the sets T_i satisfy property (i).

Observe that, by construction, $G[T_i]$ is connected. Moreover, the only edges of G that are not edges of G_{C_i} are the edges of the cycle C_i. Thus,

the only edges in the cut $(T_i, G \setminus T_i)$ of G that are not edges of the cut F_i in G_{C_i} are the edges of C_i incident with the vertices of $T_i \cap V(C_i)$. Furthermore, for every vertex w of $T_i \cap V(C_i)$, the edge wv_i belongs to the cut F_i in G_{C_i}, but not to the cut $(T_i, G \setminus T_i)$ in G. Hence, the number of edges of the cut $(T_i, G \setminus T_i)$ in G is at most $|F_i| + 2|T_i \cap V(C_i)| - |T_i \cap V(C_i)|$. Since F_i and $T_i \cap V(C_i)$ both have order at most 3, it follows that the cut $(T_i, G \setminus T_i)$ in G has order at most 6. We have therefore proved that properties (iii)-(v) hold for the sets $T_i, i \in [6r]$.

We may now define the set X. We first start with the set of (original) vertices $w_{p,q}$ of the wall, with $6r + 1 - 36(\lceil\sqrt{6r}\rceil + 1) \leq p \leq 6r + 1 + 36(\lceil\sqrt{6r}\rceil + 1)$ and $6r + 1 - (\lceil\sqrt{6r}\rceil + 1) \leq q \leq 6r + 1 - 73(\lceil\sqrt{6r}\rceil + 1)$. This set, denoted X_0, contains at least $72(\sqrt{6r} + 1)^2 \geq 72r + 73$ original vertices of the wall, due to $r \geq 1$. We now need the following:

Claim 2. *For every $i \in [6r]$, $|X_0 \cap T_i| \leq 72$.*

Sketch of proof of Claim 2. We prove the claim by showing that for every $i \in [6r]$ and every subset X_0' of X_0 that contains at least 73 vertices, there are 7 disjoint paths from vertices of $C_i \setminus T_i$ to vertices of X_0' in G. Together with property (iv), this will imply validity of Claim 2. Consider a subset X_0' of X_0 that contains least 73 original vertices of the wall. Observe that there must be 13 vertices that lie on the same horizontal path, or 7 vertices that lie on different horizontal paths. From there, taking into account the dimensions of the wall and the position of the vertices of C_i and X_0, it is easy to observe that there always exist vertices $y_1, \ldots, y_7$ in $C_i \setminus T_i$ and $x_1, \ldots, x_7$ in X_0' such that there are 7 disjoint paths between $y_1, \ldots, y_7$ and $x_1, \ldots, x_7$. $\diamond$

Therefore, the set $X_0 \cap \bigcup T_i$ contains at most $72r$ vertices, which implies that there exists a subset X of X_0 containing at least 73 vertices such that $X \cap T_i = \emptyset$ for every $i \in [6r]$. This proves property (vi) for the sets $T_i, i \in [6r]$. The validity of property (vii) follows from arguments similar to those given in the proof of Claim 2.

Finally, we show how to select sets $S_1, \ldots, S_r$ among $T_1, \ldots, T_{6r}$ so that property (ii) holds, namely that for every $1 \leq i \neq j \leq r, z_i \notin S_j$. In order to find such sets, we proceed as follows: let H be a directed graph such that $V(H) = \{T_1, \ldots, T_{6r}\}$, and (T_i, T_j) is an arc of H if and only if $z_i \in S_j$. We now claim that vertices of H have indegree at most 6. This is shown by combining properties (iv), (vi), and (vii). Assume for contradiction that there is a vertex in H having indegree at least 7, then there exist distinct indices $i_1, \ldots, i_7$ and j such that $z_{i_1}, \ldots, z_{i_7} \in S_j$. However, we know that there exist 7 disjoint paths from $\{z_{i_1}, \ldots, z_{i_7}\}$ to X

by property (vii). Together with property (vi), we obtain a contradiction with property (iv). Therefore, we conclude that the directed graph H has maximum indegree at most 6. Thus, $|E(H)| \leq 36r$, which implies that the average degree of H is at most 6. Hence, H is 6-degenerate and thus contains an independent set of size at least $\frac{|V(H)|}{6} = r$. The vertices of such an independent set correspond to sets $T_{i_1}, \ldots, T_{i_r}$ such that, for every $1 \leq p \neq q \leq r$, $z_{i_p} \notin T_{i_q}$. Therefore, we choose $S_p := T_{i_p}$ for every $p \in [r]$ and observe that the set $S_1, \ldots, S_r$ as defined indeed satisfy property (ii).

Finally, since every set T_i satisfies properties (i) and (iii)-(vi), and for every $j \in [r]$ there exists $i \in [6r]$ such that $S_j = T_i$, we obtain that the sets S_i satisfy these properties as well. This concludes the proof of the lemma. □

Lemma 2 essentially states that large treewidth yields a large number of vertex disjoint cycles that are highly connected to each other, and an additional disjoint set that is highly connected to these cycles. However, this, together with the assumption that W_4 does not immerse in G, implies that there cannot be a large flow between a vertex of degree at least 4 and one of the cycles. We will combine this fact with the notion of important separators to obtain Lemma 3.

Definition 3. *Let $X, Y \subseteq V(G)$ be vertices, $S \subseteq E(G)$ be an (X, Y)-separator, and let R be the set of vertices reachable from X in $G \setminus S$. We say that S is an important (X, Y)-separator if it is inclusion-wise minimal and there is no (X, Y)-separator S' with $|S'| \leq |S|$ such that $R' \subset R$, where R' is the set of vertices reachable from X in $G \setminus S'$.*

Theorem 4. *[4, 20] Let $X, Y \subseteq V(G)$ be two sets of vertices in graph G, let $k \geq 0$ be an integer, and let S_k be the set of all (X, Y)-important separators of size at most k. Then $|S_k| \leq 4^k$ and S_k can be constructed in time $|S_k| \cdot n^{O(1)}$.*

Theorem 4 states that the number of important separators of a certain size is bounded. The next lemma combines this fact with Lemma 2.

Lemma 3. *Let G be a graph such that G does not contain W_4 as an immersion, has no internal 3-edge cut and has a vertex u with $d(u) \geq 4$. Then the treewidth of G is upper bounded by a constant.*

Proof. If G has treewidth at least $2^{18(6r)^2 \log(6r)}$ for $r \geq 60000$, then there exist sets $Z = \{z_1, \ldots, z_r\}$, $S_1, \ldots, S_r$ and X that satisfy the properties of Lemma 2. Recall that F is an important (u, X)-separator if there is no

(u, X)-separator F' such that $|F'| \leq |F|$ and the connected component of $G - F$ that contains u is properly contained in the connected component of $G - F'$ that contains u. Additionally, observe that for every set S_i, there is an important separator F or order at most 6 such that S_i lies in the same connected component as $\{u\}$ in $G - F$. Moreover, for any cut F of order at most 6 such that S_i is contained in the same connected component as u in $G - F$, there cannot be 7 disjoint paths from u to X through F. Combined with property (vii) of Lemma 2 and the fact that every set S_i contains a vertex z_i, this implies that for every important separator F, there are at most 6 sets $S_{i_1}, \ldots, S_{i_p}, p \leq 6$, that are contained in the same connected component as u in $G - F$. However, Theorem 4 ensures that there are at most 4^6 important $(X, \{u\})$-separators of size at most 6 in G. Therefore, if $r \geq 60000 > 6 \cdot 4^6$, there is a set S_i such that the cut $(S_i, G - S_i)$ has order at least 7. Thus, we conclude that either G has an internal edge cut of order at most 3, or it has no vertex of degree at least 4, or it contains W_4 as an immersion. Hence the lemma holds. □

We are now ready to prove the main theorem of our paper.

Theorem 5. *Let G be a graph that does not contain W_4 as an immersion. Then the prime graphs of a decomposition of G via i-edge-sums, $i \in [3]$, are either subcubic graphs, or have treewidth upper bounded by a constant.*

Proof. Let us consider a decomposition of G via i-edge-sums, $i \in [3]$, and let H be a prime graph of such a decomposition. Note first that, since G does not contain W_4 as an immersion, then H does not contain it either, due to Theorem 3. Now, assume that H is not subcubic. Then there is a vertex u of degree at least 4 in H. Moreover, it is clear from Theorem 3 that H is internally 4 edge-connected. Hence, we may apply Lemma 3 and conclude that H has treewidth at most $2^{2^{13} \cdot 3^6 \cdot 5^8 \cdot \log(2^6 \cdot 3^2 \cdot 5^4)}$. Thus, the theorem holds. □

We conclude this section by noting that Theorem 5 is in a sense tight: indeed, both the fact that we decompose along edge-sums of order at most 3 and the requirement that a unique vertex of degree at least 4 is sufficient to enforce small treewidth are necessary. The fact that decomposing along internal 3-edge-sums is necessary can be seen from the fact that there are internally 3 edge-connected graphs that have vertices of degree at least 4 and yet do not contain W_4 as an immersion, e.g., a cycle where every edge is doubled.

5 Concluding remarks

Following the proof of Theorem 5, the first task is to improve the bound on the treewidth of internally 4 edge-connected graphs that exclude W_4 as an immersion and have a vertex of degree at least 4. Our proof of Theorem 5 relies on the fact that large treewidth ensures the existence of a large number of vertex disjoint cycles that are highly connected to each other. In order to obtain these cycles, we use the fact that graphs of large treewidth contain a large wall as a topological minor. However, the value of treewidth required to find a sufficiently large wall is currently enormous. Avoiding to rely on the existence of a large wall would be an efficient way to drastically reduce the constants in Lemma 2 and Theorem 5.

Another question that we leave open is to prove a similar result for larger wheels, i.e., W_k for $k \geq 5$. Providing a decomposition theorem for larger wheel seems to be a challenging task, as edge-sums no longer seem to be the proper way to proceed, since, as argued in Section 4, k edge connectivity is necessary, but W_k-immersion is not preserved under edge-sums of order $k-1$, as seen in Figure 2 and 3.

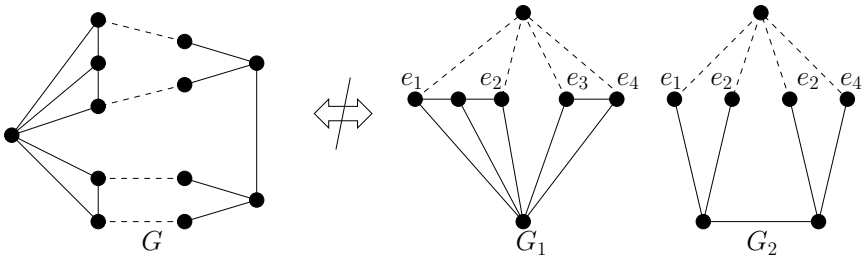


Figure 2: G_1 contains W_5 as an immersion but $G = G_1 \hat{\oplus}_4 G_2$ does not. The unique vertex in G_i incident with dotted edges is the vertex v_i, and the edge-sum maps to each other edges of G_1 and G_2 with the same label.

Decomposition theorems exist when small wheels are excluded as topological minors [9,27], however these results do not apply when excluding wheels as immersions, as in this case we must consider multigraphs. A similar important question is to characterize graphs excluding K_5 as an immersion.

Finally, note that the general algorithm to test immersion containment runs in cubic time for every fixed target graph H. We believe that Theorem 5 can be used to devise efficient algorithms to recognize graphs that exclude W_4 as an immersion. A direct application of the construction of Theorem 5 implies that this can be done in time $\tilde{O}(n^2)$. However,

we believe that this can be further improved.

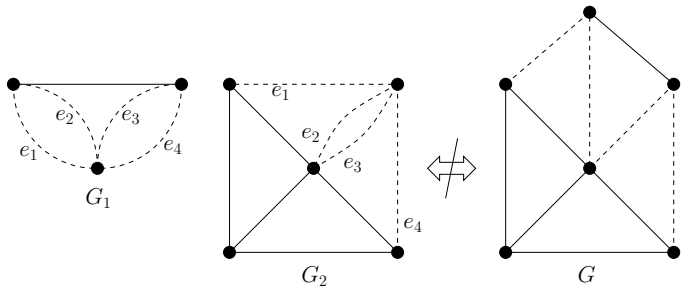


Figure 3: Neither G_1 nor G_2 contain W_4 as immersion but $G = G_1 \hat{\oplus}_4 G_2$ does. The unique vertex in G_i incident with dotted edges is the vertex v_i, and the edge-sum maps to each other edges of G_1 and G_2 with the same label.

References

[1] Adler, I., Kolliopoulos, S. G., Krause, P. K., Lokshtanov, D., Saurabh, S. and Thilikos, D. M.: Tight bounds for linkages in planar graphs. In: Proceedings of ICALP 2011, LNCS, vol. 6755, pp. 110–121, Springer, Berlin (2011)

[2] Belmonte, R. van 't Hof, P., Kamiński, M., Paulusma, D. and Thilikos, D. M.: Characterizing graphs of small carving-width. In: Proceedings of COCOA 2012. LNCS, vol.7402, pp 360–370, Springer, Berlin (2012)

[3] Bodlaender, H., Fomin, F. V., Lokshtanov, D., Penninkx, E., Saurabh, S. and Thilikos, D.M.: (Meta) Kernelization. In: Proceedings of FOCS 2009, pp. 629–638, IEEE Computer Society (2009)

[4] Chen, J., Liu, Y., and Lu S.: An improved parameterized algorithm for the minimum node multiway cut problem. In: Proceedings of WADS 2007, LNCS, vol. 4619, pp. 495–506, Springer, Berlin (2007)

[5] Dawar, A., Grohe, M. and Kreutzer, S.: Locally excluding a minor. In: Proceedings of LICS 2007, pp. 270–279, IEEE Computer Society (2007)

[6] Demaine, E. D., Fomin, F. V., Hajiaghayi, M. and Thilikos, D. M.: Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893 (2005)

[7] DeVos, M., Dvořák, Z., Fox, J., McDonald, J., Mohar, B. and Scheide, D.: Minimum degree condition forcing complete graph immersion. *ArXiv e-prints*, abs/1101.2630 (2011)

[8] Diestel R.: *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, electronic edition (2005)

[9] Farr, G.: The subgraph homeomorphism problem for small wheels. *Discrete Mathematics*, 71(2):129–142 (1988)

[10] Ferrara, M., Gould, R. J, Tansey, G. and Whalen,T.: On H-immersions. *Journal of Graph Theory*, 57(3):245–254 (2008)

[11] Giannopoulou, A. C., Kaminski, M. and Thilikos, D. M.: Forbidding Kuratowski graphs as immersions. *Arxiv e-prints*, abs/1207.5329 (2012)

[12] Giannopoulou, A. C., Salem, I. and Zoros, D.: Effective computation of immersion obstructions for unions of graph classes. In: Proceedings of SWAT 2012, LNCS, vol. 7357, pp. 165–176, Springer, Berlin (2012)

[13] Grohe, M., Kawarabayashi, K., Marx, D. and Wollan, P.: Finding topological subgraphs is fixed-parameter tractable. In: Proceedings of STOC 2011, pp. 479–488, ACM, New York (2011)

[14] Grohe, M. and Marx, D.: Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In: Proceedings of STOC 2012, pp. 173–192, ACM, New York (2012)

[15] Kawarabayashi, K., Reed, B. A. and Wollan, P.: The graph minor algorithm with parity conditions. In: Proceedings of FOCS, pp. 27–36, IEEE Computer Society (2011)

[16] Kawarabayashi, K. and Wollan, P.: A shorter proof of the graph minor algorithm: the unique linkage theorem. In: Proceedings of STOC 2010, pp. 687–694, ACM, New York (2010)

[17] Kawarabayashi, K. and Kobayashi, Y.: Linear min-max relation between the treewidth of H-minor-free graphs and its largest grid. In: Proceedings of STACS 2012, LIPIcs, vol. 14, pp. 278–289, Schloss Dagstuhl, Leibniz-Zentrum fuer Informatik (2012)

[18] Kawarabayashi, K. and Kobayashi, Y.: The induced disjoint path problem. In: Proceedings of IPCO 2008, LNCS, vol. 5035, pp. 47–61. Springer, Berlin (2008)

[19] A. Leaf and P. Seymour. Treewidth and planar minors. unpublished manuscript, 2012.

[20] Marx, D.: Parameterized graph separation problems. *Theoretical Compututer Science*, 351(3):394–406 (2006)

[21] Robertson, N. and Seymour, P. D.: Graph Minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114 (1986)

[22] Robertson, N. and Seymour, P. D.: Graph Minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110 (1995)

[23] Robertson, N. and Seymour, P. D.: Graph Minors. XXI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 89(1):43–76 (2003)

[24] Robertson, N. and Seymour, P. D.: Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357 (2004)

[25] Robertson, N. and Seymour, P. D.: Graph Minors. XXIII. Nash-Williams' immersion conjecture. *Journal of Combinatorial Theory, Series B*, 100(2):181–205 (2010)

[26] Robertson, N., Seymour, P. D. and Thomas, R.: Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348 (1994)

[27] Robinson, R. and Farr, G.: Structure and recognition of graphs with no 6-wheel subdivision. *Algorithmica*, 55(4):703–728 (2009)

[28] Seymour, P. D. and Thomas, R.: Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33 (1993)

[29] Wollan, P.: The structure of graphs not admitting a fixed immersion. *Arxiv e-prints*, abs/1302.3867 (2013)

Chapter 10

Characterizing Graphs of Small Carving-Width

Characterizing Graphs of Small Carving-Width

Rémy Belmonte[1,] Pim van 't Hof[1,†] Marcin Kamiński[2]
Daniël Paulusma[3]
Dimitrios M. Thilikos[4]

[1] Department of Informatics, University of Bergen, Norway
{remy.belmonte,pim.vanthof}@ii.uib.no

[2] Département d'Informatique, Université Libre de Bruxelles, Belgium
Institute of Computer Science, University of Warsaw, Poland
mjk@mimuw.edu.pl

[3] School of Engineering and Computing Sciences, Durham University, UK
daniel.paulusma@durham.ac.uk

[4] Department of Mathematics, National & Kapodistrian University of Athens, Greece
sedthilk@math.uoa.gr

Abstract

We characterize all graphs that have carving-width at most k for $k = 1, 2, 3$. In particular, we show that a graph has carving-width at most 3 if and only if it has maximum degree at most 3 and treewidth at most 2. This enables us to identify the immersion obstruction set for graphs of carving-width at most 3.

1 Introduction

All graphs considered in this paper are finite and undirected, have no self-loops but may have multiple edges. A graph that has no multiple edges is called *simple*. For undefined graph terminology we refer to the textbook of Diestel [7]. A *carving* of a graph G is a tree T whose internal vertices all have degree 3 and whose leaves correspond to the vertices of G. For every edge e of T, deleting e from T yields exactly two trees, whose leaves define a bipartition of the vertices of G; we say that the edge cut in G corresponding to this bipartition is *induced by* e. The *width* of a carving T is the maximum size of an edge cut in G that is induced by an edge of T. The *carving-width* of G is the minimum width of a carving of G.

Carving-width was introduced by Seymour and Thomas [17], who proved that checking whether the carving-width of a graph is at most k is an NP-complete problem. In the same paper, they proved that there is a polynomial-time algorithm for computing the carving-width of planar graphs. Later, the problem of constructing carvings of minimum width was studied by Khuller [12], who presented a polynomial-time algorithm for constructing a carving T whose width is within a $O(\log n)$ factor from the optimal. In [20] an algorithm was given that decides, in $f(k) \cdot n$ steps, whether an n-vertex graph G has carving-width at most k and, if so, also outputs a corresponding carving of G. We stress that the values of $f(k)$ in the complexity of the algorithm in [20] are huge, which makes the algorithm highly impractical even for trivial values of k.

A graph G contains a graph H as an immersion if H can be obtained from some subgraph of G after lifting a number of edges (see Section 2 for the complete definition). Recently, the immersion relation attracted a lot of attention both from the combinatorial [1, 6, 9, 21] and the algorithmic [10, 11] point of view. It can easily be observed (cf. [20]) that carving-width is a parameter closed under taking immersions, i.e., the carving-width of a graph is not smaller than the carving-width of any of its immersions. Combining this fact with the seminal result of Robertson and Seymour in [15] stating that graphs are well-quasi-ordered with respect to the immersion relation, it follows that the set $\mathcal{G}_k$ of graphs with carving-width at most k can be completely characterized by forbidding a finite set of graphs as immersions. This set is called an *immersion obstruction set* for the class $\mathcal{G}_k$.

Identifying obstruction sets is a classic problem in structural graph theory, and its difficulty may vary, depending on the considered graph class. While obstructions have been extensively studied for parameters that are closed under minors (see [2, 5, 8, 13, 14, 16, 18, 19] for a sample of such results), no obstruction characterization is known for any immersion-closed graph class. In this paper, we make a first step in this direction.

The outcome of our results is the identification of the immersion obstruction set for $\mathcal{G}_k$ when $k \leq 3$; the obstruction set for the non-trivial case $k = 3$ is depicted in Figure 3. Our proof for this case is based on a combinatorial result stating that $\mathcal{G}_3$ consists of exactly the graphs with maximum degree at most 3 and treewidth at most 2. A direct implication of our results is a linear-time algorithm for the recognition of the class $\mathcal{G}_k$ when $k = 1, 2, 3$. This can be seen as a "tailor-made" alternative to the general algorithm of [20] for elementary values of k.

2 Preliminaries

Let $G = (V, E)$ be a graph, and let $S \subset V$ be a subset of vertices of G. Then the set of edges between S and $V \setminus S$, denoted by $(S, V \setminus S)$, is an *edge cut* of G. Let the vertices of G be in 1-to-1 correspondence to the leaves of a tree T whose internal vertices all have degree 3. The correspondence between the leaves of T and the vertices of G uniquely defines the following edge weighting w on the edges of T. Let $e \in E_T$, and let C_1 and C_2 be the two connected components of $T - e$. Let S_i be the set of leaves of T that are in C_i for $i = 1, 2$; note that $S_2 = V \setminus S_1$. Then the weight $w(e)$ of the edge e in T is the number of edges in the edge cut (S_1, S_2) of G. The tree T is called a *carving* of G, and (T, w) is a *carving decomposition* of G. The *width* of a carving decomposition (T, w) is the maximum weight $w(e)$ over all $e \in E_T$. The *carving-width* of G, denoted by $\mathrm{cw}(G)$, is the minimum width over all carving decompositions of G. We define $\mathrm{cw}(G) = 0$ if $|V| = 1$. We refer to Figure 4 for an example of a graph and a carving decomposition.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{X}$ is a collection of subsets of V, called *bags*, and $\mathcal{T}$ is a tree whose vertices, called *nodes*, are the sets of $\mathcal{X}$, such that the following three properties are satisfied:

(i) for each $u \in V$, there is a bag $X \in \mathcal{X}$ with $u \in X$;

(ii) for each $uv \in E$, there is a bag $X \in \mathcal{X}$ with $u, v \in X$;

(iii) for each $u \in V$, the nodes containing u induce a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{T}, \mathcal{X})$ is the size of a largest bag in $\mathcal{X}$ minus 1. The *treewidth* of G, denoted by $\mathrm{tw}(G)$, is the minimum width over all possible tree decompositions of G.

Let $G = (V, E)$ be a graph, and let uv be an edge of G. The *contraction* of the edge uv is the operation that deletes u and v from G and replaces them by a new vertex x that is made adjacent to the neighbors of u and of v in G, such that for every vertex $w \in V \setminus \{u, v\}$, the number of edges between x and w in the new graph is equal to the number of edges between w and $\{u, v\}$ in G. A graph G contains a graph H as a *minor* if H can be obtained from G by a sequence of vertex deletions, edge deletions and edge contractions. The following two well-known properties of treewidth will be used in the proof of our main result.

Lemma 1 (cf. [4])**.** *Let G be a simple graph and k an integer. If $\mathrm{tw}(G) \leq k$, then G contains a vertex of degree at most k.*

Lemma 2 (cf. [4, 7])**.** *Let G be a graph. Then $tw(H) \leq tw(G)$ for every minor H of G.*

The *subdivision* of an edge uv is the operation that deletes the edge uv from the graph and adds a new vertex w as well as two new edges uw and vw. The reverse operation is called *vertex dissolution*; this operation removes a vertex v of degree 2 that has two distinct neighbors u and w, and adds a new edge between u and w, regardless of whether or not there already exist edges between u and w. A graph G contains a graph H as a *topological minor* if H can be obtained from G by a sequence of vertex deletions, edge deletions, and vertex dissolutions. Equivalently, G contains H as a topological minor if G contains a subgraph H' that is a *subdivision* of H, i.e., H' can be obtained from H by a sequence of edge subdivisions. The following lemma is obtained by combining some well-known properties of treewidth, minors, and topological minors.

Lemma 3 (cf. [7])**.** *A graph has treewidth at most 2 if and only if it does not contain K_4 as a topological minor.*

Let u, v, w be three distinct vertices in a graph such that uv and vw are edges. The operation that removes the edges uv and vw, and adds the edge uw (even in the case u and w are already adjacent) is called a *lift*. A graph G contains a graph H as an *immersion* if H can be obtained from G by a sequence of vertex deletions, edge deletions, and lifts. Note that dissolving a vertex v of degree 2 that has two distinct neighbors u and w is equivalent to first lifting the edges uv and vw and then deleting the vertex v. Hence, it readily follows from the definitions of topological minors and immersions that every topological minor of graph G is also an immersion of G.

3 The Main Result

We begin this section by stating some useful properties of carving-width. The following observation is known and easy to verify by considering the number of edges in the edge cut $(\{u\}, V \setminus \{u\})$ of a graph $G = (V, E)$.

Observation 1. *Let G be a graph. Then* $\mathrm{cw}(G) \geq \Delta(G)$.

We also need the following two straightforward lemmas. The first lemma follows from the observation that any subgraph of a graph is an immersion of that graph, combined with the observation that carving-width is a parameter that is closed under taking immersions (cf. [20]). We include the proof of the second lemma for completeness.

Lemma 4. *Let G be a graph. Then* $\mathrm{cw}(H) \leq \mathrm{cw}(G)$ *for every subgraph H of G.*

Lemma 5. *Let G be a graph with connected components $G_1, \ldots, G_p$ for some integer $p \geq 1$. Then* $\mathrm{cw}(G) = \max\{\mathrm{cw}(G_i) \mid 1 \leq i \leq p\}$.

Proof. Since the carving-width of a graph with only one vertex is defined to be 0, the lemma clearly holds if G has no edges. Suppose G has at least one edge. Lemma 4 implies that $\max\{\mathrm{cw}(G_i) \mid 1 \leq i \leq p\} \leq \mathrm{cw}(G)$. Now let (T_i, w_i) be a carving decomposition of G_i of width $\mathrm{cw}(G_i)$ for $i = 1, \ldots, p$. Since deleting isolated vertices does not change the carving-width of a graph, we may without loss of generality assume that G has no isolated vertices. In particular, this means that each tree T_i contains at least one edge. We construct a carving decomposition (T, w) of G from the p carving decompositions (T_i, w_i) as follows.

We pick an arbitrary edge $e_i = x_i y_i$ in each T_i. For each $i \in \{2, \ldots, p-1\}$, we subdivide the edge e_i twice by replacing it with edges $x_i z_i$, $z_i z_i'$ and $z_i' y_i$, where z_i and z_i' are two new vertices. The edges e_1 and e_p are subdivided only once: the edge e_1 is replaced with a new vertex z_1 and two new edges $x_1 z_1$ and $z_1 y_1$, and the edge e_p is replaced with a new vertex z_p' and two new edges $x_p z_p'$ and $z_p' y_p$. Finally, we add the edge $z_i z_{i+1}'$ for each $i \in \{1, \ldots, p-1\}$. This results in a tree T whose internal vertices all have degree 3. Since there are no edges between any two connected components of G, the corresponding carving decomposition (T, w) of G has width $\max\{\mathrm{cw}(G_i) \mid 1 \leq i \leq p\}$. Hence, $\mathrm{cw}(G) \leq \max\{\mathrm{cw}(G_i) \mid 1 \leq i \leq p\}$. We conclude that $\mathrm{cw}(G) = \max\{\mathrm{cw}(G_i) \mid 1 \leq i \leq p\}$. □

The next lemma is the final lemma we need in order to prove our main result.

Lemma 6. *Let G' be a graph with carving-width at least 2, and let uw be an edge of G'. Let G be the graph obtained from G' by subdividing the edge uw. Then* $\mathrm{cw}(G) = \mathrm{cw}(G')$.

Proof. Let (T', w') be a carving decomposition of G' of width $\mathrm{cw}(G') \geq 2$, and let p be the unique neighbor of u in T'. Let v be the vertex that was used to subdivide the edge uw in G', i.e., the graph G was obtained from G' by replacing uw with edges uv and vw for some new vertex v. Let T be the tree obtained from T' by first relabeling the leaf in T' corresponding to vertex u by q, and then adding two new vertices u and v as well as two new edges qu and qv; see Figure 1 for an illustration. Let us show that the resulting carving decomposition (T, w) of G has width at most $\mathrm{cw}(G')$.

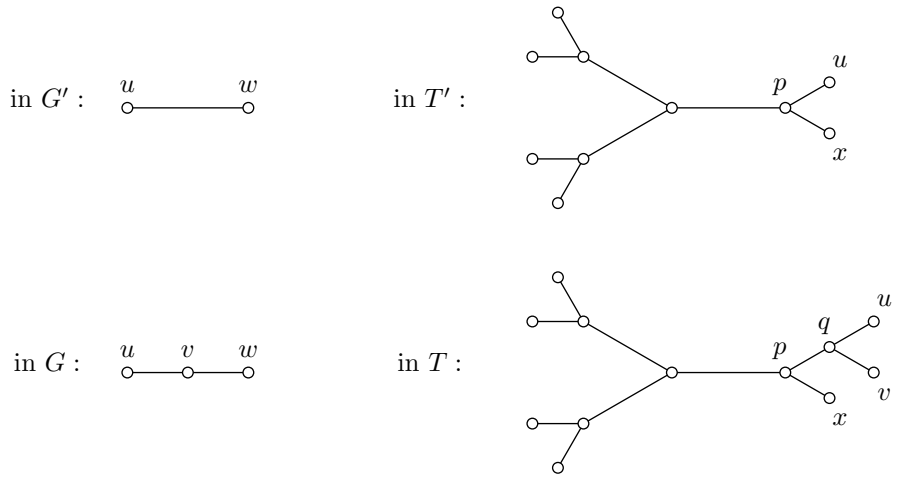


Figure 1: A schematic illustration of how the tree T' in the carving decomposition of G' is transformed into a tree T in the proof of Lemma 6 when the edge uw in G' is subdivided. The vertex x is an arbitrary vertex of G', possibly w.

Let e be an edge in T. Suppose that $e = pq$. By definition, $w(e)$ is the number of edges between $\{u, v\}$ and $V \setminus \{u, v\}$ in G, which is equal to the number of edges incident with u in G'. The latter number is the weight of the edge pu in T'. Hence, $w(e) \leq \text{cw}(G')$. Suppose that $e = qu$. By definition, $w(e)$ is the number of edges incident with u in G, which is equal to the number of edges incident with u in G'. Hence $w(e) \leq \text{cw}(G')$. Suppose that $e = qv$. By definition, $w(e)$ is the number of edges incident with v in G, which is 2. Hence $w(e) = 2 \leq \text{cw}(G')$. Finally, suppose that $e \notin \{pq, qu, qv\}$. Let C_1 and C_2 denote the subtrees of T obtained after removing e. Let S_i be the set of leaves of T in C_i for $i = 1, 2$. Then u and v either both belong to S_1 or both belong to S_2. Without loss of generality, assume that both u and v belong to S_1. By definition, $w(e)$ is the number of edges between S_1 and S_2 in G, which is equal to the number of edges between $S_1 \setminus \{v\}$ and S_2 in G'. The latter number is the weight of the edge e in T'. Hence, $w(e) \leq \text{cw}(G')$. We conclude that (T, w) has width at most $\text{cw}(G')$, and hence $\text{cw}(G) \leq \text{cw}(G')$.

It remains to show that $\text{cw}(G) \geq \text{cw}(G')$. Let (T^*, w^*) be a carving decomposition of G of width $\text{cw}(G)$. We remove the leaf corresponding to v from T^*. Afterwards, the neighbor of v in T^* has degree 2, and we dissolve this vertex. This results in a tree T''. It is easy to see that the corresponding carving decomposition (T'', w'') of G' has width

at most $\mathrm{cw}(G)$. Hence, $\mathrm{cw}(G) \geq \mathrm{cw}(G')$. This completes the proof of Lemma 6. □

We are now ready to show the main result of our paper.

Theorem 1. *Let G be a graph. Then the following three statements hold.*

(i) $\mathrm{cw}(G) \leq 1$ *if and only if* $\Delta(G) \leq 1$.

(ii) $\mathrm{cw}(G) \leq 2$ *if and only if* $\Delta(G) \leq 2$.

(iii) $\mathrm{cw}(G) \leq 3$ *if and only if* $\Delta(G) \leq 3$ *and* $\mathrm{tw}(G) \leq 2$.

Proof. Let $G = (V, E)$ be a graph. By Lemma 5 we may assume that G is connected. We prove the three statements separately.

(i) If $\mathrm{cw}(G) \leq 1$, then $\Delta(G) \leq 1$ due to Observation 1. If $\Delta(G) \leq 1$, then G is isomorphic to either K_1 or K_2. Clearly, $\mathrm{cw}(G) \leq 1$ in both cases.

(ii) If $\mathrm{cw}(G) \leq 2$, then $\Delta(G) \leq 2$ due to Observation 1. If $\Delta(G) = 1$, then $\mathrm{cw}(G) \leq 1$ follows from (i). If $\Delta(G) = 2$, then G is either a graph consisting of two vertices with two edges between them, or a simple graph that is either a path or a cycle. In all three cases, it is clear that $\mathrm{cw}(G) \leq 2$.

(iii) First suppose that $\mathrm{cw}(G) \leq 3$. Then $\Delta(G) \leq 3$ due to Observation 1. We need to show that $\mathrm{tw}(G) \leq 2$. For contradiction, suppose that $\mathrm{tw}(G) \geq 3$. Then, by Lemma 3, G contains K_4 as a topological minor, i.e., G contains a subgraph H such that H is a subdivision of K_4. Since $\mathrm{cw}(K_4) = 4$, we have that $\mathrm{cw}(H) = \mathrm{cw}(K_4) = 4$ as a result of Lemma 6. Since H is a subgraph of G, Lemma 4 implies that $\mathrm{cw}(G) \geq \mathrm{cw}(H) = 4$, contradicting the assumption that $\mathrm{cw}(G) \leq 3$.

For the reverse direction, we need to prove that every graph $G = (V, E)$ with $\Delta(G) \leq 3$ and $\mathrm{tw}(G) \leq 2$ has carving-width at most 3. We use induction on $|V|$. If $|V| \leq 2$, then G is either isomorphic to K_1 or K_2, or G consists of two vertices with exactly two edges between them. It is clear that $\mathrm{cw}(G) \leq 3$ in each of these cases. From now on, we assume that $|V| \geq 3$.

First, suppose that G contains two vertices u and v with at least two edges between them. Since $|V| \geq 3$, we may without loss of generality assume that v has a neighbor $t \neq u$. Then, because $\Delta(G) \leq 3$ and there are at least two edges between u and v in G, we find that t and u are the only two neighbors of v in G and that the number of edges between u and v is exactly 2. Let G^* denote the graph obtained from G by deleting one edge between u and v, and let G' denote the graph obtained from G^* by

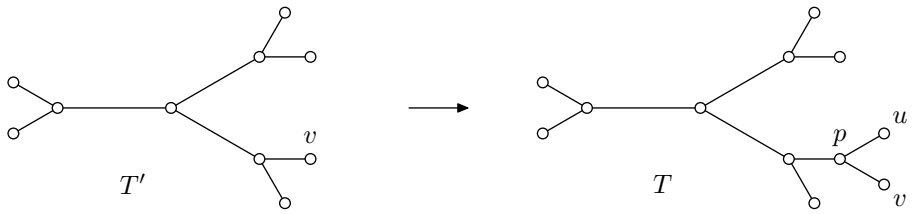


Figure 2: A schematic illustration of how the tree T is constructed from the tree T' in the proof of Theorem 1.

dissolving v. Note that G' is a connected graph on $|V| - 1$ vertices, and $\Delta(G') \leq 3$. Moreover, since G' is a topological minor and hence a minor of G, Lemma 2 ensures that $\mathrm{tw}(G') \leq \mathrm{tw}(G) \leq 2$. Consequently, we can apply the induction hypothesis to deduce that $\mathrm{cw}(G') \leq 3$.

If $\mathrm{cw}(G') \leq 1$, then $\Delta(G') \leq 1$ by Observation 1. Since G' is a connected graph on $|V|-1 \geq 2$ vertices, G' must be a path on two vertices. Then G^* is a path on three vertices, implying that $\mathrm{cw}(G^*) = 2$. Since G can be obtained from G^* by adding a single edge, $\mathrm{cw}(G) \leq 3$ in this case. Suppose $2 \leq \mathrm{cw}(G') \leq 3$. Then, by Lemma 6, $\mathrm{cw}(G^*) = \mathrm{cw}(G') \leq 3$. Moreover, from the proof of Lemma 6 it is clear that there exists a carving decomposition (T^*, w^*) of G^* of width $\mathrm{cw}(G^*)$ such that u and v have a common neighbor q in T^*. We consider the carving decomposition (T, w) of G with $T = T^*$. Let e be an edge in T. First suppose that $e = uq$ or $e = vq$. Then $w(e) \leq 3$, as both u and v have degree at most 3 in G. Now suppose that $e \notin \{uq, vq\}$. Then $w(e) = w^*(e) \leq \mathrm{cw}(G^*) \leq 3$. We conclude that the carving decomposition (T, w) of G has width at most 3, which implies that $\mathrm{cw}(G) \leq 3$.

From now on, we assume that G contains no multiple edges, i.e., we assume that G is simple. Since $\mathrm{tw}(G) \leq 2$, G contains a vertex of degree at most 2 due to Lemma 1.

Suppose G contains a vertex u of degree 1. Let v be the neighbor of u in G, and let G' be the graph obtained from G by deleting u. It is clear that $\Delta(G') \leq 3$ and $\mathrm{tw}(G') \leq 2$, so $\mathrm{cw}(G') \leq 3$ by the induction hypothesis. Let (T', w') be a carving decomposition of G' of width $\mathrm{cw}(G')$. Let T be the tree obtained from T' by first changing the label of the leaf of T' corresponding to vertex v into p, and then adding two new vertices u and v and two new edges pu and pv; see Figure 2. Since v is the only neighbor of u in G, it is easy to see that the width of the resulting carving decomposition (T, w) of G is at most $\mathrm{cw}(G')$, which implies that $\mathrm{cw}(G) \leq \mathrm{cw}(G') \leq 2$.

Finally, suppose that G contains a vertex u of degree 2. Since we

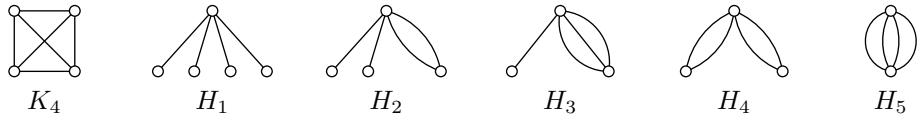


Figure 3: The immersion obstruction set for graphs of carving-width at most 3.

assume G to be simple, u has two distinct neighbors v and t. Let $G' = (V', E')$ denote the connected graph obtained from G by dissolving u. Note that G' has maximum degree at most 3, and that $\text{tw}(G') \leq 2$ due to Lemma 2 and the fact that G' is a minor of G. Hence, by the induction hypothesis, $\text{cw}(G') \leq 3$. If $\text{cw}(G') \leq 1$, then $\Delta(G') \leq 1$ by Observation 1. This, together with the observation that G' is a connected graph on $|V| - 1 \geq 2$ vertices, implies that G' is a path on two vertices. Consequently, G is a path on three vertices, and hence $\text{cw}(G) = 2 \leq 3$. If $2 \leq \text{cw}(G') \leq 3$, then we can apply Lemma 6 to conclude that $\text{cw}(G) = \text{cw}(G') \leq 3$. This completes the proof of Theorem 1. □

Since graphs of treewidth at most 2 can easily be recognized in linear time, Theorem 1 implies a linear-time recognition algorithm for graphs of carving-width at most 3.

Thilikos, Serna and Bodlaender [20] proved that for any k, there exists a linear-time algorithm for constructing the immersion obstruction set for graphs of carving-width at most k. For $k \in \{1, 2\}$, finding such a set is trivial. We now present an explicit description of the immersion obstruction set for graphs of carving-width at most 3.

Corollary 1. *A graph has carving-width at most* 3 *if and only if it does not contain any of the six graphs in Figure 3 as an immersion.*

Proof. Let G be a graph. We first show that if G contains one of the graphs in Figure 3 as an immersion, then G has carving-width at least 4. In order to see this, it suffices to observe that the graphs $K_4, H_1, \ldots, H_5$ all have carving-width 4. Hence, G has carving-width at least 4, because carving-width is a parameter that is closed under taking immersions (cf. [20]).

Now suppose that G has carving-width at least 4. Then, due to Theorem 1, $\Delta(G) \geq 4$ or $\text{tw}(G) \geq 3$. If $\Delta(G) \geq 4$, then G has a vertex v of degree at least 4. By considering v and four of its incident edges, it is clear that G contains one of the graphs $H_1, \ldots, H_5$ as a subgraph, and consequently as an immersion. If $\text{tw}(G) \geq 3$, then Lemma 3 im-

plies that G contains K_4 as a topological minor, and consequently as an immersion. □

From the proof of Corollary 1, we can observe that an alternative version of Corollary 1 states that a graph has carving-width at most 3 if and only if it does not contain any of the six graphs in Figure 3 as a topological minor.

4 Conclusions

Extending Theorem 1 to higher values of carving-width remains an open problem, and finding the immersion obstruction set for graphs of carving-width at most 4 already seems to be a challenging task. We proved that for any graph G, $\text{cw}(G) \leq 3$ if and only if $\Delta(G) \leq 3$ and $\text{tw}(G) \leq 2$. We finish our paper by showing that the equivalence "$\text{cw}(G) \leq 4$ if and only if $\Delta(G) \leq 4$ and $\text{tw}(G) \leq 3$" does not hold in either direction.

To show that the forward implication is false, we consider the pentagonal prism F_5, which is displayed in Figure 4 together with a carving decomposition (T, w) of width 4. Hence, $\text{cw}(F_5) \leq 4$. However, F_5 is a minimal obstruction for graphs of treewidth at most 3 [4], implying that $\text{tw}(F_5) = 4$.

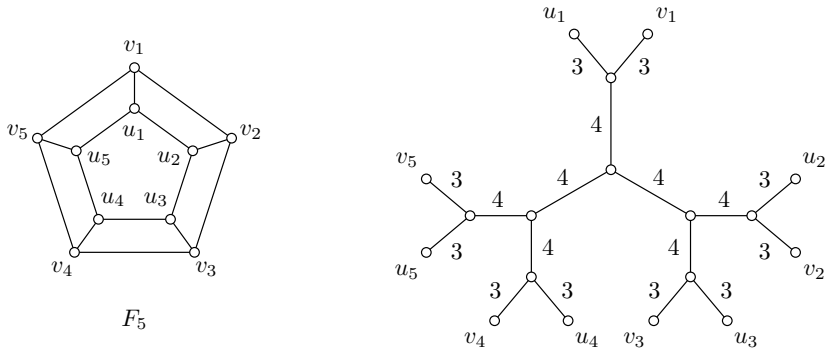


Figure 4: The pentagonal prism F_5 and a carving decomposition (T, w) of F_5 that has width 4.

To show that the backward implication is false, we consider the graph K_5^-, which is the graph obtained from K_5 by removing an edge. Note that $\Delta(K_5^-) = 4$ and $\text{tw}(K_5^-) = 3$. It is not hard to verify that $\text{cw}(K_5) = 6$. Since removing an edge decreases the carving-width by at most 1, we conclude that $\text{cw}(K_5^-) \geq 5$.

References

[1] Abu-Khzam, F.N., Langston, M.A.: Graph coloring and the immersion order. In: Warnow, T., Zhu, B. (eds.) COCOON 2003. LNCS, vol. 2697, pp. 394–403. Springer (2003)

[2] Arnborg, S., Proskurowski, A.: Characterization and recognition of partial 3-trees. SIAM J. Algebraic Discrete Methods 7(2), 305–314 (1986)

[3] Belmonte, R., van 't Hof, P., Kamiński, M., Paulusma, D., Thilikos, D.M.: Characterizing graohs of small carving-width. In: Lin, G. (ed.) COCOA 2012. LNCS, vol. 7402, pp. 360–370. Springer (2012)

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

[5] Bodlaender, H.L., Thilikos, D.M.: Graphs with branchwidth at most three. Journal of Algorithms 32(2), 167–194 (1999)

[6] DeVos, M., Dvořák, Z., Fox, J., McDonald, J., Mohar, B., Scheide, D.: Minimum degree condition forcing complete graph immersion. *CoRR*, arXiv:1101.2630, January 2011.

[7] Diestel, R.: Graph Theory. Springer-Verlag, Electronic Edition (2005)

[8] Dvořák, Z., Giannopoulou, A.C., Thilikos, D.M.: Forbidden graphs for tree-depth. European Journal of Combinatorics 33(5), 969–979 (2012)

[9] Giannopoulou, A.C., Kamiński, M., Thilikos, D.M.: Forbidding Kuratowski graphs as immersions. Submitted for publication.

[10] Grohe, M., Kawarabayashi, K., Marx, D., Wollan, P.: Finding topological subgraphs is fixed-parameter tractable. In: Proceedings of STOC 2011, pp. 479–488, ACM (2011)

[11] Kawarabayashi, K., Kobayashi, Y.: List-coloring graphs without subdivisions and without immersions. In: Proceedings of SODA 2012, pp. 1425–1435, SIAM (2012)

[12] Khuller, S., Raghavachari, B., Young, N.: Designing multicommodity flow trees. Information Processing Letters 50, 49–55 (1994)

[13] Koutsonas, A., Thilikos, D.M., Yamazaki, K.: Outerplanar obstructions for matroid pathwidth. Electronic Notes in Discrete Mathematics 38, 541–546 (2011)

[14] Robertson, N., Seymour, P.D., Thomas, R.: Linkless embeddings of graphs in 3-space. Bull. Amer. Math. Soc. 28, 84–89 (1993)

[15] Robertson, N., Seymour, P.D.: Graph minors XXIII. Nash-Williams' immersion conjecture. Journal of Combinatorial Theory, Series B 100, 181–205 (2010)

[16] Rué, J., Stavropoulos, K.S., Thilikos, D.M.: Outerplanar obstructions for a feedback vertex set. European Journal of Combinatorics 33, 948–968 (2012)

[17] Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. Combinatorica 14(2), 217–241 (1994)

[18] Takahashi, A., Ueno, S., Kajitani, Y.: Minimal acyclic forbidden minors for the family of graphs with bounded path-width. Discrete Mathematics 127, 293–304 (1994)

[19] Thilikos, D.M.: Algorithms and obstructions for linear-width and related search parameters. Discrete Applied Mathematics 105, 239–271 (2000)

[20] Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Constructive linear time algorithms for small cutwidth and carving-width. In: Lee, D.T., Teng, S.-H. (eds.) ISAAC 2000. LNCS, vol. 1969, pp. 192–203. Springer (2000)

[21] Wollan, P.: The structure of graphs not admitting a fixed immersion. Submitted for publication. Preprint available at `http://arxiv.org/abs/1302.3867`.