TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Tree-Decomposition

Graph Minor Theory and Algorithmic Implications

Ausgeführt am Institut für

Diskrete Mathematik und Geometrie

der Technischen Universität Wien

unter Anleitung von

Univ.Prof. Dipl.-Ing. Dr.techn. Michael Drmota

durch

Miriam Heinz, B.Sc.

Matrikelnummer: 0625661

Baumgartenstraße 53
1140 Wien

Datum

Unterschrift

Preface

The focus of this thesis is the concept of *tree-decomposition*. A tree-decomposition of a graph G is a representation of G in a tree-like structure. From this structure it is possible to deduce certain connectivity properties of G. Such information can be used to construct efficient algorithms to solve problems on G. Sometimes problems which are $\mathcal{NP}$-hard in general are solvable in polynomial or even linear time when restricted to trees. Employing the tree-like structure of tree-decompositions these algorithms for trees can be adapted to graphs of bounded *tree-width*. This results in many important algorithmic applications of tree-decomposition.

The concept of tree-decomposition also proves to be useful in the study of fundamental questions in graph theory. It was used extensively by Robertson and Seymour in their seminal work on Wagner's conjecture. Their *Graph Minor*s series of papers spans more than 500 pages and results in a proof of the *graph minor theorem*, settling Wagner's conjecture in 2004. However, it is not only the proof of this deep and powerful theorem which merits mention. Also the concepts and tools developed for the proof have had a major impact on the field of graph theory. Tree-decomposition is one of these spin-offs. Therefore, we will study both its use in the context of graph minor theory and its several algorithmic implications.

A complete survey of the many theoretical and practical applications of tree-decomposition would go beyond the scope of this thesis. We therefore chose several interesting aspects making no claim to be exhaustive.

The first chapter contains a brief introduction to graph theory and summarises the basic definitions and concepts needed for this thesis. Main source for this chapter was the textbook on graph theory by Diestel [16].

In the second chapter we take a closer look at the graph minor theorem and its graph theoretic context. We introduce the most important terms such as minors and well-quasi-orders and state the graph minor theorem. The results presented are mainly taken from [16, 35, 37]. Two theorems by Kuratowski [34] and Kruskal [33] and a proof technique created by Nash-Williams [37] give insight into the topic of graph minor theory.

Tree-decomposition is discussed in detail in the third chapter. First, we introduce the concepts of tree-decomposition and tree-width. We also explain the connectivity properties a graph G shares with its tree-decompositions [16, 41]. Then we examine several notions closely related to tree-decomposition. A different representation of a graph G of bounded tree-width can give insight into different properties of G. We discuss clique-sums, (partial) k-trees and chordal graphs. For sources, see [11, 12, 16, 35, 49, 50],

amongst others. Last, we present a few excluded minor theorems employing some of the notions defined before. Here, we refer to [18, 30].

The intention of the fourth chapter is to present an example how tree-decomposition is employed to prove the graph minor theorem. Hence we discuss one of the main parts of the proof, namely the exclusion of a planar graph H. We describe the class of graphs not containing H as a minor. The results of this chapter can be combined to prove a weaker version of the graph minor theorem. The main sources are [17] and [23].

The fifth chapter is dedicated to some of the many algorithmic applications of tree-decomposition. We introduce the k disjoint paths problem which is closely related to the problem of H minor containment. Both problems have already been shown to be solvable in polynomial time by Robertson and Seymour [45]. We present an improved version of their algorithm [29]. Then we discuss the usage of tree-decomposition for solving complex problems on graphs of bounded tree-width. We explain a generic approach utilising a given tree-decomposition [10]. As a last example we introduce a pursuit-evasion game on graphs studied by Seymour and Thomas [53].

The sixth chapter gives some additional information about related research and open problems.

Apart from [16], the introductory surveys [8], [30], and [35] provided useful information for most of the topics discussed in this thesis.

Acknowledgements

I want to thank my supervisor, Professor Dr. Michael Drmota, for giving me the opportunity to write my master thesis in the field of graph theory and for sparking my interest in graph minor theory. I want to further thank him for his guidance, encouragement and patience.

Furthermore, I want to thank my father for his constant support – for always being there for me and believing in me.

This thesis is dedicated to my parents.

Contents

Chapter 1

Preliminaries

This chapter is a brief introduction to some basic graph theoretic terms and concepts used in this thesis. It should also avoid possible notational confusion. Most definitions are standard and can be found in textbooks like [16] or [14]. In case of discrepancies we will mostly refer to [16]. However, a basic knowledge of graph theory is assumed throughout this thesis.

1.1 Graph structure

An undirected *graph* $G = (V, E)$ consists of a set V of *vertices* and a set E of *edges* between these vertices, also denoted by $V(G)$ and $E(G)$, respectively. We denote an edge $e \in E(G)$ between two vertices $u, v \in V(G)$ by $\{u, v\}$ or simply uv (or vu as orientation does not matter). e *joins* u and v and u, v are called *endvertices* of e. A *loop* is an edge vv.

Furthermore, a graph can have *multiple edges* between two vertices and more than one loop at a vertex. We call a graph a *multigraph* if loops and multiple edges are allowed, otherwise *simple graph*.

In case of simple graphs, E is usually seen as a subset of $[V]^2$, the set of all subsets of V with cardinality 2. For multigraphs this definition is not sufficient, with multiple edges a multiset E would be more suitable. [16, p. 28], for example, defines a map $E \mapsto V \cup [V]^2$ to assign endvertices to each edge in E. Since it is not crucial for this work, in this thesis E will be seen as a multiset. As aforementioned, an edge will be represented by its endvertices, even if with this representation uniqueness may be lost.

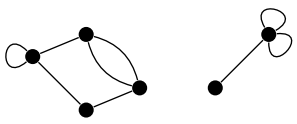

Figure 1.1: undirected graph with loops and multiple edges

A vertex v is *incident* to an edge e if v is an endvertex of e. Equivalently, e is called *incident* to v. We will also write $v \in e$. The *vertex degree* $d_G(v)$, or simply $d(v)$, of a vertex v is the number of edges incident to v, loops counting twice. We say that G has

maximum degree $\Delta(G)$ if $\Delta(G)$ is the maximum over all vertex degrees of vertices in G. Two vertices $u, v \in V$ are *adjacent* if there exists an edge $uv \in E$. u and v are called *neighbours* and

$$N(v) = N_G(v) = \{w \in V \mid \exists e \in E : e = vw\}$$

is the *neighbourhood* of v in G. In case of loops v can be its own neighbour. The neighbourhood of a set $W \subseteq V$ is defined by $N(W) = \bigcup_{w \in W} N(w) \setminus W$. A vertex set $W \subseteq V$ where no two vertices are adjacent is a *stable* set of vertices in G.

The *order* of a graph $G = (V, E)$ is the cardinality $|V|$ of its vertex set. A *finite* graph has order $|V| = n \in \mathbb{N}$. If not otherwise declared, the graphs in this thesis will be finite undirected multigraphs.

A *complete graph* is a graph G where any two vertices $u, v \in V(G)$ are adjacent. In a complete simple graph with n vertices every vertex has degree $d(v) = n - 1$. We denote this graph by K_n.

$G = (V, E)$ is called *bipartite* if its vertex set can be partitioned into two sets V_1, V_2 such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ and $\nexists e \in E : e \subseteq V_1 \vee e \subseteq V_2$ holds. In other words, the vertices of a bipartite graph can be coloured with two colours, such that the endvertices of each edge do not have the same colour. A (simple) bipartite graph G is called *complete* if no new edge can be added to G without losing its property of being bipartite. The *complete bipartite graph* $K_{m,n}$ is the complete bipartite graph with $|V_1| = m$ and $|V_2| = n$.

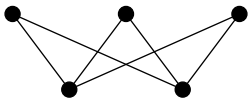

Figure 1.2: complete bipartite graph $K_{3,2}$

1.2 Graph operations and subgraphs

There are several ways to modify a graph obtaining a new graph. We present different operations on graphs and possible relations between two graphs.

In the following, let $G = (V, E)$ be the graph considered.

Definition 1.2.1 (edge deletion) Deleting an edge results in a graph with same vertex set V and edge set $E \setminus \{e\}$. This graph is denoted by $G - e$. In general, for a set of edges F, the graph $G - F = (V, E \setminus F)$ is obtained.

Definition 1.2.2 (vertex deletion) If a vertex v is deleted it is necessary to delete all edges incident to v as well. Thus, we have $G - v = (V \setminus \{e\}, E \setminus \{e \in E \mid v \in e\})$ and, more generally, $G - W = (V \setminus W, E \setminus \{e \in E \mid \exists v \in W : v \in e\})$ for $W \subseteq V$.

Definition 1.2.3 (edge contraction) Contraction of an edge $e = uv$ yields a graph G/e with a new vertex v_e that replaces e. u and v are deleted and v_e is adjacent to all their neighbours. More formally, every edge $uw \in E$ or $vw \in E$ is replaced by an edge $v_e w \in E(G/e)$. Edges not incident to u or v remain unchanged. In a multigraph multiple edges may become loops. Loops can not be contracted, only deleted.

Definition 1.2.4 (subdivision of an edge) Subdivision of an edge $e = uv$ inserts a new vertex w of degree 2 into e that divides e into uw and wv. More generally, e can be replaced by a uv-path whose inner vertices all have degree 2.

Definition 1.2.5 (suppression of a vertex) Suppressing a vertex is the inversion of subdividing an edge, that is, deleting a vertex of degree 2 and adding an edge between its neighbours. In the case of multiple edges, this can create loops. A vertex with a loop is simply deleted.

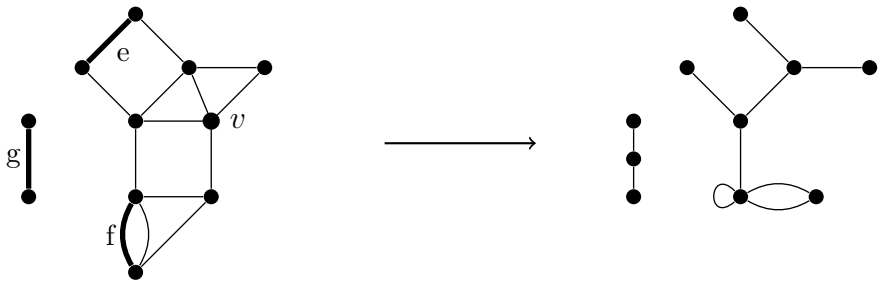

Figure 1.3: deletion of edge e and vertex v, contraction of f and subdivision of g

Definition 1.2.6 (subgraph) A *subgraph* $G' = (V', E')$ of G is a graph with $V' \subseteq V$ and $E' \subseteq E$. We write $G' \subseteq G$. G is a *supergraph* of G'.

Definition 1.2.7 (induced subgraph) A subgraph $G' = (V', E')$ of G is *induced by* $W \subseteq V$ if $V' = W$ and $E' = \{uv \in E \mid u, v \in W\}$. The subgraph of G induced by the vertex set W is denoted by $G[W]$. Analogously, we define the subgraph $G[F]$ of G induced by a set $F \subseteq E$ of edges as $G[F] = (\{v \in V \mid \exists e \in F : v \in e\}, F)$.

Remark 1.2.8 *Note that an edge-induced subgraph cannot contain isolated vertices (vertices with degree 0).*

Definition 1.2.9 (spanning subgraph) If $V(G') = V$, a subgraph G' of G is called *spanning subgraph.*

Definition 1.2.10 (clique) A maximal complete subgraph C of a graph G is called a *clique* of G. That means that C is not contained in any other complete subgraph of G of higher order. The *clique number* $\omega(G)$ is the maximal order taken over all cliques of G.

Definition 1.2.11 (union, intersection and difference of graphs) The *union* of two graphs G and G' is a graph with vertex set $V(G) \cup V(G')$ and edge set $E(G) \cup E(G')$. Their *intersection* $G \cap G' = (V(G) \cap V(G'), E(G) \cap E(G')$ is defined similarly.

Two graphs possibly intersect. Their *difference* $G - G'$, or $G \setminus G'$, denotes the graph obtained from G by deleting all vertices and edges which are contained in G' as well.

Definition 1.2.12 Two subgraphs are called *disjoint* if they have no vertex in common. Otherwise they are *meeting.*

1.3 Connectivity

For the rest of this section let $G = (V, E)$ be a graph and $u, v \in V$ two vertices of G.

A *path* P_G^{uv} from u to v, or *uv-path*, is a (finite) sequence of distinct vertices and edges

$$v_0 e_0 v_1 e_1 \dots v_{n-1} e_{n-1} v_n$$

of G with $v_0 = u$, $v_n = v$ and $e_i = v_i v_{i+1}$ for $i = 0, \dots, n-1$. u and v are the *endvertices* of P_G^{uv}, the other vertices are called *inner vertices*. A path has to have at least one vertex. The *length* of a path is its number of edges. Sometimes it is sufficient to note P_G^{uv} as a sequence of vertices, if the choice of edges is clear from the context or not important. Also the index G can be omitted if the reference is clear.

A path P in G is also seen as a subgraph of G consisting exactly of the vertices and edges from its defining sequence. We will mostly use this interpretation. Therefore, we will use graph operations on paths as well.

Two paths *meet* if they have at least one vertex in common. They are *(internally) disjoint* if the sets of their (internal) vertices are disjoint.

A path P^{uw} between two sets $U, W \subseteq V$ with $U \cap P^{uv} = \{u\}$ and $W \cap P^{uv} = \{w\}$ is called UV-path. In case of $U = \{u\}$ or $W = \{w\}$ we write uW-path or Uw-path, respectively. A *cycle* is a uv-path with the only difference that u equals v. We define a cycle to have at least one edge. Therefore a path consisting of only one vertex is not a cycle.

A graph is called *connected* if there exists an uv-path for every two vertices u, v. A *connected component* of a graph G is a maximal connected subgraph G' of G. Maximal meaning that no vertex or edge of $G - G'$ can be added to G' without losing the property of connectedness. The number of connected components of a graph G is denoted by $c(G)$.

A *tree* is a connected graph without cycles, a *forest* a graph whose connected components are trees. For two vertices u, v in a tree T there always exists one unique path between them. We denote it by T^{uv}. Sometimes it is useful to mark one vertex of a tree T as special, the *root* $r(T)$, or simply r, of the tree. Since there is a unique path between the root and any vertex of T, we can define a partial order on the set of vertices of T by

$$u \leq_T v \iff u \in T^{rv}.$$

A rooted tree can also be interpreted as a directed graph, in which all edges have a *head* and a *tail* and are directed away from the root. That way every vertex has exactly one *predecessor*, except for the root, which has none. A vertex with no *successor* is called *leaf* of T. The term leaf is also used for vertices of degree 1 in undirected trees. Let $v \in V(T) \setminus r(T)$ be an arbitrary vertex and e the edge whose head is v. We call the connected component of $T - e$ containing v the *branch* of T at v. It is a tree with root v and its vertices and edges are ordered and directed as they are in T.

If a spanning subgraph of a graph G is a tree it is called a *spanning tree* of G. If G is not connected but every connected component has a spanning tree the set of these trees is a *spanning forest* of G.

An edge $e \in E$ whose deletion increases the number of connected components of G is called a *bridge*. In a tree every edge is a bridge. Analogously, a vertex with the same

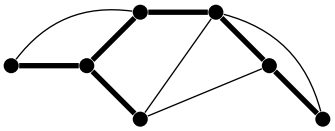

Figure 1.4: graph with a spanning tree (marked by thick edges)

property is called *cutvertex*. In the case of multigraphs a vertex with a loop is seen as a cutvertex. More generally, a *separator* is a set $S \subseteq V$ whose deletion increases the number of connected components. We say, S *separates* G. A set $S \subseteq V$ *separates* two sets of vertices $U, W \in V$ if $U \setminus S$ and $W \setminus S$ lie in different components of $G - S$. A graph $G = (V, E)$ is called *n-connected* if $G - W$ is connected for all $W \subseteq V$ with $|W| < n$, provided that $n < |V|$. The *connectivity* of G is the greatest $k \in \mathbb{N}$ such that G is k-connected. It is denoted by $\kappa(G)$, $\kappa(G) \geq 1$ meaning that G is connected, $\kappa(G) = 0$ that G is disconnected. The *vertex connectivity* $\kappa_G(u, v)$ between two vertices u and v in G equals $|S| - 1$, where $S \subseteq V$ is a set of minimum cardinality such that S separates $\{u\}$ from $\{v\}$. G is *n-edge-connected* if $n < |E|$ and $G - F$ is connected for all $F \subseteq E$ with $|F| < n$. $\lambda(G)$ is the greatest integer n for which G is n-edge-connected, the *edge-connectivity* of G.

A famous theorem of Menger explains the relation between separating sets and disjoint paths in a graph.

Theorem 1.3.1 (Menger's theorem) *Let $G = (V, E)$ be a graph and $U, W \subseteq V$ two vertex sets. Then the minimum number of vertices separating U from W in G is equal to the maximum number of disjoint UW-paths in G.*

Proof. G has to contain at least as many vertices separating U from W as there exist disjoint UW-paths, at least one vertex from each path.

To show the converse we use induction on the number $|E|$ of edges. For $|E| = 0$, $U \cap W$ is a set of minimal cardinality separating U from W and we also have $U \cap W$ trivial UW-paths, each consisting of exactly one vertex.

Now let G contain at least one edge $e = xy$ and let s be the minimum number of vertices separating U from W in G. Assume that there are at most $l < s$ disjoint UW-paths in G. As a consequence, G/e also contains at most l disjoint $U'W'$-paths, where U' and W' differ from U and W inasmuch as they contain v_e if U respectively W contains an endvertex of e. By induction hypothesis, there is a set $S' \subseteq V(G/e)$ of cardinality at most l separating U' from W'. S' has to contain v_e, otherwise it would be an UW-separating set in G. $S = S' \setminus v_e \cup \{x, y\}$ is an UW-separating set in G and therefore we have $|S| \geq s$ and

$$l + 1 \geq |S| \geq s \geq l + 1.$$

Furthermore, we can easily construct $s = l + 1$ disjoint UW-paths. Look at $G - e$. Every set separating U from S also separates U from W and therefore contains at least s vertices. Thus we can find s disjoint US-paths and analogously s disjoint SW-paths in $G - e$. As $|S| = s$ and these paths can only meet in S, we can combine them to s disjoint UW-paths in G which contradicts our choice of l. □

Chapter 2

Graph Minor Theory

Tree-decompositions are extensively used in the context of graph minor theory. Introduced by Robertson and Seymour within their Graph Minors series of papers [41] the concept of tree-decomposition proved to be useful in many ways. The Graph Minors series spans over more than 20 papers and more than 500 pages altogether. Its main purpose was to prove an important open conjecture, often called Wagner's conjecture.

In this chapter we introduce the basic terms needed to state this conjecture and some of their properties. Then we give a brief overview of the context behind Wagner's conjecture and its relation to minor-closed classes of graphs. We also state some well-known results by Kuratowski [34] and Kruskal [33] which are closely related to Wagner's conjecture. The definitions and results presented in this chapter can be found in [16, 35, 37].

2.1 Well-quasi-order and minors

Definition 2.1.1 (isomorphism) An *isomorphism* between two graphs G and H is a map $f : V(G) \to V(H)$ such that an edge uv is in $E(G)$ if and only if the edge $f(u)f(v)$ is in $E(H)$. Regarding multigraphs, also the number of multiple edges and loops has to coincide. Two graphs G, H are called *isomorphic*, denoted by $G \simeq H$, if there exists an isomorphism between them.

Usually, we speak of a graph meaning its isomorphism class. For example, we speak of *the* complete bipartite graph $K_{m,n}$. But for the discussion of graph minor theory, we will not identify two isomorphic graphs and always make clear if a (minor) relation is defined up to isomorphism.

Definition 2.1.2 (graph invariant) A *graph invariant* is a function which, taking graphs as input, assigns the same value to isomorphic graphs. Examples are functions giving the order of a graph or stating if it is simple/connected/loop-free.

We have already introduced the subgraph relation between two graphs. A subgraph H of G arises from edge and vertex deletion. Similarly, we will define subdivisions and (topological) minors.

Definition 2.1.3 (subdivision) A graph H is a *subdivision* of a graph G if H is obtained from G by subdivision of edges.

Definition 2.1.4 (topological minor) A *topological minor* H of G is obtained from G by deletion of edges and vertices and suppression of vertices.

If a graph H is isomorphic to a topological minor of a graph G we say G *topologically contains* H.

Definition 2.1.5 (minor) If not only deletion of edges and vertices but also contraction of edges is allowed the resulting graph is called a *minor* of the initial graph. A minor of G that is not isomorphic to G is called a *proper minor* of G.

We will denote the minor-relation "H is isomorphic to a minor of G" by $H \lesssim G$ and say that H is a $\lesssim$-minor of G.

A topological minor H of G is also a minor of G, since suppression of a vertex is the same as contracting an edge with an endvertex of degree 2. The converse is not necessarily true.

Definition 2.1.6 A class of graphs $\mathcal{G}$ is called *minor-closed* if it contains for every graph in $\mathcal{G}$ all its $\lesssim$-minors as well.

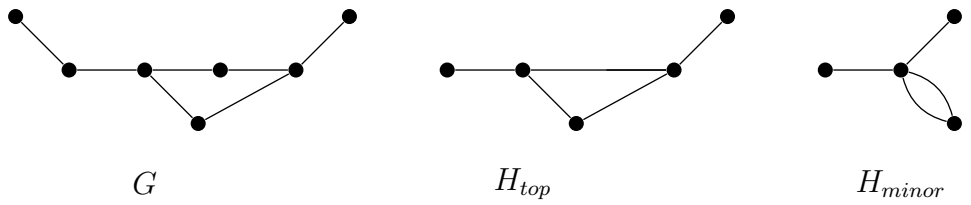

Figure 2.1: graph G with topological minor H_{top} and minor H_{minor}

Definition 2.1.7 (well-quasi-order) A *quasi-order* is a reflexive and transitive relation $\leq$. A quasi-ordered set $(S, \leq)$ is *well-quasi-ordered* if in every infinite sequence $s_0, s_1, s_2 \ldots$ of elements of S there exist indices i, j such that $i < j$ and $s_i \leq s_j$. According to [37], we call such a pair (s_i, s_j) a *good pair* of the sequence. An infinite sequence is *good* if it contains at least one good pair, otherwise it is a *bad* sequence.

Lemma 2.1.8 *A quasi-ordered set $(S, \leq)$ is well-quasi-ordered if and only if there exists neither an infinite strictly descending chain nor an infinite antichain of elements of S.*

Proof. $\Rightarrow$: By definition of well-quasi-order, S cannot have an infinite strictly descending chain nor an infinite antichain.

$\Leftarrow$: Conversely, let $s_0, s_1, s_2, \ldots$ be an infinite sequence of elements of S. Assume it is a bad sequence.

Let $S_{s_0\geq}$ denote the set of all $s_j \in s_1, s_2, \ldots$ with $s_0 \geq s_j$. $S_{s_0\geq}$ has to be finite because S contains neither an infinite strictly descending chain nor an infinite antichain. So the subsequence $t_1, t_2, \ldots$ of $s_1, s_2, \ldots$, containing only the elements not in $S_{s_0\geq}$, consists only of elements incomparable to s_0. Choose a maximal subsequence $(t_0 = t_{i_0}), t_{i_1}, t_{i_2}, \ldots$

of $t_1, t_2, \ldots$ where each i_j is the smallest possible index such that $t_{i_0}, \ldots, t_{i_{j-1}}, t_{i_j}$ is an antichain. This sequence itself is an antichain and therefore finite. Let i_n be the maximal occurring index. The union $\bigcup_{j=0}^{n} S_{t_{i_j}\geq}$ is a finite union of finite sets and therefore finite which is a contradiction to $s_0, s_1, s_2, \ldots$ being infinite. □

Definition 2.1.9 Given an infinite sequence $s_0, s_1, s_2, \ldots$ of elements of a quasi-ordered set $(S, \leq)$, we call an element s_i *terminal* if there exists no s_j with $i < j$ and $s_i \leq s_j$.

Lemma 2.1.10 *If $(S, \leq)$ is well-quasi-ordered the number of terminal elements in an infinite sequence is finite. Furthermore, every infinite sequence of elements of S has an infinite ascending subsequence.*

Proof. Infinitely many terminal elements would contain either an infinite strictly descending chain or an infinite antichain which is contradictory to S being well-quasi-ordered.

Thus, for an arbitrary infinite sequence $s_0, s_1, s_2, \ldots$, there is an index N such that all terminal elements of this sequence have an index smaller than N. So, starting with s_{i_0}, $i_0 \geq N$, we can choose s_{i_1} such that $i_0 < i_1$ and $s_{i_0} \leq s_{i_1}$. We continue with s_{i_2} such that $s_{i_1} \leq s_{i_2}$, $i_1 < i_2$ and by analogously adding elements we create an infinite ascending subsequence $s_{i_0}, s_{i_1}, s_{i_2}, \ldots$. □

Lemma 2.1.11 *If $(S, \leq_S)$ and $(T, \leq_T)$ are well-quasi-ordered sets then $S \times T$ is well-quasi-ordered by the relation $\leq$ defined by*

$$(s_1, t_1) \leq (s_2, t_2) \iff s_1 \leq_S s_2 \wedge t_1 \leq_T t_2 \, .$$

Proof. Let $(s_0, t_0), (s_1, t_1), (s_2, t_2), \ldots$ be an infinite sequence of elements of $S \times T$. Since S is well-quasi-ordered we can find an infinite ascending subsequence $s_{i_0}, s_{i_1}, s_{i_2} \ldots$ of $s_0, s_1, s_2, \ldots$. The sequence $t_{i_0}, t_{i_1}, t_{i_2} \ldots$ has to contain two elements t_k and t_l with $t_k \leq_T t_l$ and $k < l$, as T is well-quasi-ordered. So we found elements (s_k, t_k), (s_l, t_l) such that $(s_k, t_k) \leq (s_l, t_l)$ and $k < l$. □

For an ordered set $(S, \leq_S)$ we denote the class of finite subsets of S by $[S]^{<\omega}$ and define a quasi-order $\leq_\omega$ on $[S]^{<\omega}$ by

$$A \leq_\omega B \iff \exists \text{ injective mapping } f : A \to B \text{ such that } a \leq_S f(a) \; \forall a \in A.$$

f is called a *non-descending* mapping.

Lemma 2.1.12 *If $(S, \leq)$ is well-quasi-ordered then $([S]^{<\omega}, \leq_\omega)$ is well-quasi-ordered.*

Proof. Assume $[S]^{<\omega}$ is not well-quasi-ordered.

Consider a sequence $S_0, S_1, S_2, \ldots$ such that every S_i, $i \geq 0$ is a set with minimal cardinality such that $S_0, S_1, \ldots, S_i$ are the first members of a bad sequence in $[S]^{<\omega}$. This sequence itself is a bad sequence and contains only non-empty subsets of S. Set $T_i = S_i \setminus \{s_i\}$, for $i \geq 0$ and arbitrary elements $s_i \in S_i$. There cannot exist a bad sequence consisting of sets T_i since then there would be a (bad) subsequence $T_{i_0}, T_{i_1}, T_{i_2}, \ldots$ with $i_0 \leq i_j$ for all j. As a consequence, the sequence

$$S_1, S_2, \ldots, S_{i_0 - 1}, T_{i_0}, T_{i_1}, T_{i_2}, \ldots$$

would be bad since a good pair (S_j, T_{i_j}) would imply $S_j \leq_\omega T_{i_j} \leq_\omega S_{i_j}$ for $j \leq i_j$ which is not possible. This is a contradiction to the choice of S_{i_0}. Therefore, the class $\mathcal{T}$ of sets T_i is well-quasi-ordered and by Lemma 2.1.11 also $S \times \mathcal{T}$ is well-quasi-ordered. That implies that the sequence

$$(s_0, T_0), (s_1, T_1), (s_2, T_2), \ldots$$

contains a good pair $((s_i, T_i), (s_j, T_j))$. But this would mean that $S_i \leq_\omega S_j$ which is a contradiction. □

Lemma 2.1.13 *The minor relation $\lesssim$ is a quasi-order on the class of finite graphs.*

Proof. This is easy to see since every graph is a minor of itself and a minor of a minor of G is also a minor of G. □

Identifying two isomorphic copies of a graph would make $\lesssim$ a partial order. Since an infinite strictly descending chain of $\lesssim$-minors is not possible, $\lesssim$ is a well-quasi-order if and only if no infinite antichain exists.

2.2 The graph minor theorem

We have now defined everything we need to state Wagner's conjecture. It is nowadays also known as *graph minor theorem* or *Robertson-Seymour theorem* after it was proven by Robertson and Seymour in 2004 [47]. Throughout their papers of the Graph Minors series it is referred to as Wagner's conjecture, although apparently Wagner did not state this conjecture; see [16, p. 373].

Theorem 2.2.1 (graph minor theorem) *The class of finite graphs is well-quasi-ordered by the minor relation $\lesssim$.*

To interpret this theorem in another context, we can look at graph properties that can be described by excluded minors.

Definition 2.2.2 (excluded minor) We call a graph H an *excluded minor* from a class of graphs $\mathcal{G}$ if no graph in $\mathcal{G}$ contains H as a $\lesssim$-minor.

Lemma 2.2.3 *A minor-closed class of graphs $\mathcal{G}$ can be characterised by its excluded minors.*

This can easily be achieved by taking all graphs not in $\mathcal{G}$. We can also go further and only consider the minimal elements $\mathcal{K}_\mathcal{G}$ of this set, that is, graphs with no proper minor not in $\mathcal{G}$. Conversely, by defining a set of excluded minors $\mathcal{M}$, we get a minor-closed class of graphs. Such a class will be denoted by $Forb(\mathcal{M})$, the set of all graphs not containing any of the graphs in $\mathcal{M}$ as a $\lesssim$-minor.

Now it follows from the graph minor theorem that a finite list of graphs suffices to describe a graph property that is inherited by minors.

Corollary 2.2.4 (of Theorem 2.2.1) *Every minor-closed class of graphs $\mathcal{G}$ can be described by a finite set $\mathcal{K}_\mathcal{G}$ of (pairwise incomparable) excluded minors.*

Proof. Let $\mathcal{G}$ be $Forb(\mathcal{M})$. Such a set $\mathcal{M}$ always exists, just take all graphs not in $\mathcal{G}$. If $\mathcal{M}$ is an infinite set of graphs, then by Theorem 2.2.1 a graph in $\mathcal{M}$ exists that is a minor of another graph in $\mathcal{M}$. If we now only consider graphs that are $\lesssim$-minimal in $\mathcal{M}$, we obtain a finite set $\mathcal{K}_\mathcal{G}$ of excluded minors. □

The set $\mathcal{K}_\mathcal{G}$ is sometimes called *Kuratowski set* for $\mathcal{G}$. Indeed, K. Kuratowski stated one well-known characterization theorem that can be rewritten in terms of excluded minors [34]. It applies to planar graphs.

Definition 2.2.5 (planarity) A graph G is called *planar* if it can be drawn in the plane without crossings. We also say that G is *embeddable in the plane*. A graph without this property is called *nonplanar*.

Adding loops and multiple edges to a given graph does not change its (non)planarity. Obviously, the property of planarity is inherited by minors.

Lemma 2.2.6 *The class of planar graphs $\mathcal{P}$ is minor-closed.*

Theorem 2.2.7 (Kuratowski) *A graph is planar if and only if it contains no topological minor isomorphic to the complete graph K_5 or the complete bipartite graph $K_{3,3}$.*

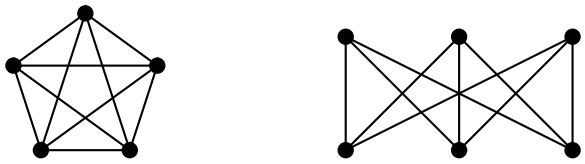

Figure 2.2: the complete graph K_5 and the complete bipartite graph $K_{3,3}$

We can formulate Theorem 2.2.7 with minors as well. Since K_5 and $K_{3,3}$ are not planar and planarity is inherited by (topological) minors, a planar graph cannot have K_5 nor $K_{3,3}$ as a (topological) minor. Conversely, if a graph does contain neither K_5 nor $K_{3,3}$ as a minor it can contain neither one of them as a topological minor. Those two graphs are both minimally nonplanar, that is, they contain no proper minor that is nonplanar. Furthermore, any other excluded minor would have to contain either one of them as a topological minor by Theorem 2.2.7. So the Kuratowski set $\mathcal{K}_\mathcal{P}$ consists only of K_5 and $K_{3,3}$.

Theorem 2.2.8 *A graph is planar if and only if it contains no minor isomorphic to the complete graph K_5 or the complete bipartite graph $K_{3,3}$.*

Kruskal proved another important minor theorem [33]. It can be interpreted as a graph minor theorem for trees.

Theorem 2.2.9 (Kruskal) *The class of finite trees is well-quasi-ordered by the relation of topological containment.*

Here, we present a proof which is due to Nash-Williams [37] and whose proof-technique will also be used in other proofs related to the graph minor theorem. For example,

their concept of finding a "minimal bad sequence" was already utilised in the proof of Lemma 2.1.12.

Now we consider rooted trees and a relation $\leq$ such that $T \leq T'$ means that there exists an injective function f mapping $V(T)$ to $V(T')$ such that

$$v \leq_T u \Leftrightarrow f(v) \leq_{T'} f(u)$$

holds for all vertices $u, v \in V(T)$. This is obviously a quasi-order. If we show that the class of finite rooted trees $\mathcal{R}$ is well-quasi-ordered by this relation $\leq$, this also implies Theorem 2.2.9.

Proof of Theorem 2.2.9. Assume $\mathcal{R}$ is not well-quasi-ordered by $\leq$. We construct a minimal bad sequence as in the proof of Lemma 2.1.12. Let $T_0, T_1, T_2, \ldots$ be such a sequence and each T_i, $i \geq 0$ be a rooted tree of minimal cardinality such that $T_0, \ldots, T_i$ is the beginning of a bad sequence of elements of $\mathcal{R}$. Now, for each T_i consider the set of branches $\mathcal{B}_i$ of T_i at all neighbours of $r(T_i)$. Let $\mathcal{B}$ be the union of all these sets $\mathcal{B}_i$. If there existed a bad sequence $B_0, B_1, B_2, \ldots$ with $B_i \in \mathcal{B}_{f(i)}$ and $f(0) \leq f(j) \forall j$, then the sequence

$$T_0, T_1, \ldots, T_{f(0)-1}, B_0, B_1, B_2, \ldots$$

would also be a bad sequence since $T_i \leq B_j$ would imply $T_i \leq T_{f(j)}$. As this would be contradictory to the choice of $T_{f(0)}$ we conclude that $\mathcal{B}$ is well-quasi-ordered. Lemma 2.1.12 assures the existence of a good pair $(\mathcal{B}_i, \mathcal{B}_j)$, $i < j$, since we can construct a sequence containing only sets of the form $\mathcal{B}_k \in \mathcal{B}^{<\omega}$. $\mathcal{B}_i \leq_\omega \mathcal{B}_j$ means that the branches of T_i can be mapped onto (different) branches of T_j. By mapping $r(T_i)$ onto $r(T_j)$, this results in $T_i \leq T_j$ and we have a contradiction. Therefore $\mathcal{R}$ is well-quasi-ordered. $\square$

Chapter 3

Tree-Decomposition

The focus of this chapter is on tree-decomposition and some of its applications in graph minor theory. In this context we will introduce tree-width which has already been studied by Halin under a different name [24]. Apparently, this went unnoticed by Robertson and Seymour who reintroduced this concept for their proof of the graph minor theorem [41].

First, we will present the notions of tree-decomposition and tree-width and some of their basic properties. Then it is shown how tree-width measures the connectivity of a graph. These properties can be found in [41, 16, 8, 32].

Second, there exist several equivalent representations of tree-decomposition. We present the concepts of clique-sums, partial k-trees and chordal graphs and explain their connection to tree-decomposition. Bodlaender discussed several characterisations in [11] and [12]. Clique-sums are defined in [30] and [35]. Partial k-trees are covered, for example, in [2, 4, 50, 55]. For chordal graphs see [16, 22, 49].

Furthermore, we show how the exclusion of some specific minor H can affect the structure of the graphs contained in $Forb(H)$. In many cases this can be expressed in the context of tree-decompositions. We present some excluded minor theorems which can be found in [30, 18, 35].

In their Graph Minors series of papers, Robertson and Seymour studied the structure of graphs not containing some planar graph G. This and the general case of excluding an arbitrary graph H play an important part in the proof of the graph minor theorem. In chapter 4, we therefore discuss the exclusion of a planar graph in detail.

3.1 Tree-decomposition

Definition 3.1.1 A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(T, \mathcal{V})$ consisting of a tree T and a family $\mathcal{V} = (V_t)_{t \in V(T)}$ of subsets of V and satisfying the following properties:

(TD1) $\bigcup_{t \in V(T)} V_t = V$

(TD2) $\forall uv \in E : \exists t \in V(T)$ with $u, v \in V_t$

(TD1) $\forall t_1, t_2, t_3 \in V(T) : t_2 \in T^{t_1 t_3} \Rightarrow V_{t_1} \cap V_{t_3} \subseteq V_{t_2}$

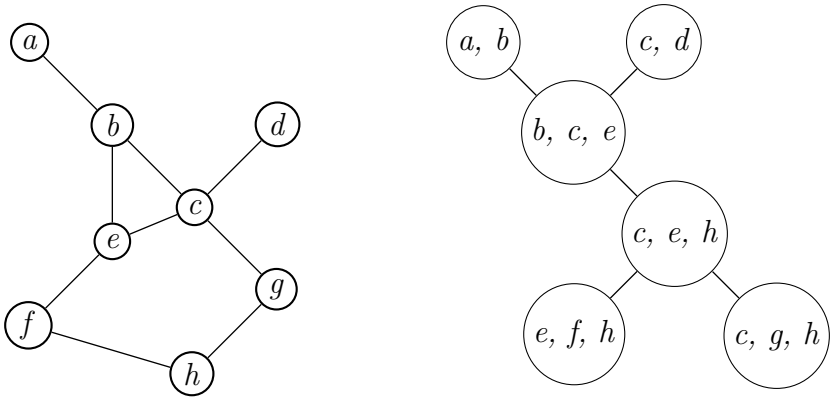

Figure 3.1: tree-decomposition of width 2

We call the sets V_t, and sometimes also their induced subgraphs $G[V_t]$, the *parts* of the tree-decomposition $(T, \mathcal{V})$. If T is a path, we call $(T, \mathcal{V})$ a *path-decomposition*. For the rest of this section, let $(T, \mathcal{V})$ be a tree-decomposition of $G = (V, E)$ as defined above.

Definition 3.1.2 The *width* of a tree-decomposition is

$$\max_{t \in V(T)} (|V_t| - 1).$$

The *tree-width* of a graph G is the minimum width over all possible tree-decompositions of G.

Tree-width is a graph invariant and describes how "tree-like" the structure of a graph is. More precisely, it gives information about the connectivity of the graph. This is due to the fact that the tree T of $(T, \mathcal{V})$ has separation properties similar to those of the corresponding graph G.

A tree-composition can also be seen as a union of subtrees, since by (TD3) the vertices of T containing a certain vertex $v \in V(G)$ form a subtree T_v of T. If E contains an edge uv then (TD2) ensures that the subtrees T_u and T_v intersect.

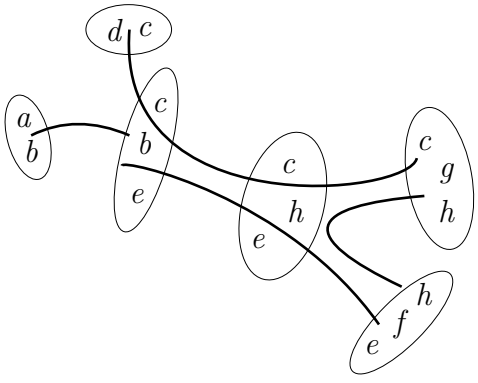

Figure 3.2: subtrees formed by vertices of $V(G)$

Remark 3.1.3 *Trees and forests have tree-width at most 1. The complete graph K_n has tree-width $n - 1$.*

Definition 3.1.4 Graphs obtained from one single edge by subdivision and by addition of parallel edges to already existing edges are called *series-parallel*. Figure 3.1 shows a tree-decomposition of a series-parallel graph.

Lemma 3.1.5 *Series-parallel graphs have tree-width at most 2.*

Proof. This can be seen by observing that K_2 has tree-width 1 and duplicating edges does not increase tree-width. And that, for a graph G' obtained from a graph G by subdivision of an edge uv by a new vertex w, a tree-decomposition $(T, \mathcal{V})$ of G can easily be transformed into a tree-decomposition for G' by adding a new part $\{u, v, w\}$ and connecting it to a part containing the edge uv. □

Lemma 3.1.6 *If H is a subgraph of a graph G then H has tree-width at most $tw(G)$.*

Proof. Let $(T, \mathcal{V})$ be a tree-decomposition of G of width $tw(G)$. We can construct from $(T, \mathcal{V})$ a tree-decomposition $(T, \mathcal{W})$ of H with parts

$$W_t = V_t \cap V(H).$$

Obviously, $(T, \mathcal{W})$ has width at most $tw(G)$. □

Lemma 3.1.7 *G has tree-width at most k if and only if all connected components of G have tree-width at most k.* □

Lemma 3.1.8 *If H is a minor of a graph G then H has tree-width at most $tw(G)$.*

Proof. As can easily bee seen, deleting vertices or edges can not increase the tree-width of a graph. Now we examine contraction of an edge. Let $(T, \mathcal{V})$ be a tree-decomposition of $G = (V, E)$ of width w and $e = uv \in E$ the edge to be contracted to a vertex v_e. Set W_i to be the part V_i, with the only difference that every occurrence of u or v is replaced by v_e and set $\bar{T} = T$. We now show that $(\bar{T}, \mathcal{W})$ is a tree decomposition of H with width at most w since replacing u and v by v_e can not increase the tree-width w.

Intuitively, it is clear that we have found a tree-decomposition of H. T_u and T_v were subtrees of T meeting in at least one vertex t. From this follows that $\bar{T}_{v_e}$ will be connected and since $\bar{T}$ contains no cycles $\bar{T}_{v_e}$ will also be a subtree of $\bar{T}$.

To be more accurate, we prove (TD1)-(TD3) for $(\bar{T}, \mathcal{W})$. (TD1) is clear from (TD1) for $(T, \mathcal{V})$ and the replacement of u, v by v_e. For every edge wu, $w \in V$, there exists t' with $w, u \in V_{t'}$ by (TD2), so $W_{t'}$ contains wv_e. The same follows for edges wv, $w \in V$, and this implies (TD2) for $(\bar{T}, \mathcal{W})$. Now let $t_1, t_2 \in E(\bar{T})$ with v_e in $W_{t_1} \cap W_{t_2}$. That means that either $u \in V_{t_1} \cap V_{t_2}$ or $v \in V_{t_1} \cap V_{t_2}$ or (without loss of generality) $u \in V_{t_1} \setminus V_{t_2}$ and $v \in V_{t_2} \setminus V_{t_1}$. The first two cases, together with (TD3) for $(T, \mathcal{V})$, imply

$$\forall t \in T^{t_1 t_2} : \; v_e \in W_t$$

for $(\bar{T}, \mathcal{W})$. Concerning the third case, we know that, by (TD2) and (TD3) for $(T, \mathcal{V})$, there has to exist $t \in T^{t_1 t_2}$ such that both u, v are in V_t. (TD3) for $(T, \mathcal{V})$ now tells us that u is a member of every set $V_{t'}$ between V_{t_1} and V_t, and v is contained in the other parts of the path $T^{t_1 t_2}$. Thus v_e is included in every $W_{t''}$ with $t'' \in \bar{T}^{t_1 t_2}$, which completes the proof. □

Bodlaender proved the following two lemmata in [11]. They are not difficult to understand but are useful for finding an upper bound for the tree-width of a given graph.

Lemma 3.1.9 *Let $G = (V, E)$ be a graph and $W \subseteq V$ a set of vertices. If $G - W$ has tree-width at most k then G has tree-width at most $k + |W|$.*

Proof. Let $(T, \mathcal{V})$ be a tree-decomposition of $G - W$ of width at most k. Then $(T, \mathcal{W})$ with parts

$$W_t = V_t \cup W,\ t \in V(T)$$

is a tree-decomposition of G of width at most $k + |W|$. □

Lemma 3.1.10 *Let $G = (V, E)$ be a graph and $F \subseteq E$ a set of edges. If $G - F$ has tree-width at most k then G has tree-width at most $k + |F|$.*

Proof. Let $V_F \subseteq$ be a set of vertices of cardinality at most $|F|$ containing at least one endvertex of every edge in F. As $G - V_F$ is a subgraph of $G - F$, we know by Lemma 3.1.6 that

$$tw(G - V_F) \leq tw(G - F) \leq k.$$

Therefore, by Lemma 3.1.9, G has tree-width at most $k + |V_F| \leq k + |F|$. □

3.2 Tree-decomposition and connectivity

Given a tree-decomposition $(T, \mathcal{V})$ of a graph G, it is possible to read off certain information about the connectivity of G. The vertices contained in a part V_t separate the sets of vertices of G characterised by the components of $T - t$. Therefore, the subgraphs of G induced by these sets can only "communicate" through V_t. So G and T have similar connectivity properties. This knowledge is used to develop fast algorithms for graphs of bounded tree-width. We will later present such algorithms.

In this section we will always assume that we have given a tree-decomposition $(T, \mathcal{V})$ of a graph G.

Definition 3.2.1 The connected components of $T - t$ are called the *branches* of T at $t \in V(T)$.

Lemma 3.2.2 *Let $t \in V(T)$ and $v \in V$, then $v \in V_t$ or v is only contained in vertex sets V_u of exactly one branch of T at t, denoted by $T_t(v)$.*

Proof. v has to be an element of one of the vertex sets $V_{t'}$, $t' \in V(T)$ by $(TD1)$. Suppose, $v \notin V_t$ and there are $t_1, t_2 \in V(T)$ with $v \in V_i$, $i = 1, 2$, that are separated by t. Then $t \in T^{t_1 t_2}$ and together with $(TD3)$ it follows that $v \in V_t$, which is a contradiction. □

Lemma 3.2.3 *If $uv \in E$ and u and v are not elements of V_t, then $T_t(u) = T_t(v)$.*

Proof. By (TD2), u and v are in $V_{t'}$ for some $t' \neq t$. This part $V_{t'}$ is in $T_t(u)$ and $T_t(v)$, which implies their equality. □

Lemma 3.2.4 *If $v, v' \in V$ are not elements of a part V_t and are not separated in G by V_t, then $T_t(v) = T_t(v')$.*

Proof. Let $P_G^{vv'} = vu_1u_2 \dots u_nv'$ be a vv'-path such that no vertex of $P_G^{vv'}$ is an element of V_t. Lemma 3.2.3 implies

$$V_t(v) = V_t(u_1) = V_t(u_2) = \dots = V_t(u_n) = V_t(v') \; . \qquad \square$$

Corollary 3.2.5 *Let $t \in V(T)$ and let $T_1, T_2, \dots, T_k$ be the branches of T at t. Denote the subgraph of G induced by the vertex set*

$$\bigcup_{l \in V(T_i)} V_l$$

by G_i, for $i = 1, \dots, k$. Then the subgraphs

$$G_1 - V_t, G_2 - V_t, \dots, G_k - V_t$$

have neither a vertex in common nor edges between them.

Lemma 3.2.6 *Let $e = t_1t_2$ be an edge of T and denote the vertex sets of the two components of $T - e$ by N_{t_1}, N_{t_2} respectively. Then $V_{t_1} \cap V_{t_2}$ separates $U_{t_1} = \bigcup_{t \in N_{t_1}} V_t$ and $U_{t_2} = \bigcup_{t \in N_{t_2}} V_t$ in G.*

Proof. For $v \in U_{t_1} \cap U_{t_2}$ it follows from (TD3) that v is also an element of V_{t_1} and V_{t_2}, therefore $v \in V_{t_1} \cap V_{t_2}$.

Let $v_1 \in U_{t_1} \setminus U_{t_2}$ and $v_2 \in U_{t_2} \setminus U_{t_1}$. Suppose that v_1 and v_2 are separated by neither V_{t_1} nor V_{t_2}. Then, by Lemma 3.2.4 we have $V_{t_1}(v_1) = V_{t_1}(v_2)$ and analogously $V_{t_2}(v_1) = V_{t_2}(v_2)$, which is a contradiction to the choice of v_1 and v_2. So every v_1v_2-path has vertices in V_{t_1} and V_{t_2}. Moreover at least one of its edges is an element of V_{t_1} or V_{t_2}, so $V_{t_1} \cap V_{t_2}$ contains at least one vertex of each v_1v_2-path. $\square$

Lemma 3.2.7 *Given a set $W \subseteq V$, there is either $t \in T$ such that $W \subseteq V_t$ or there exist $w_1, w_2 \in W$ and $t_1t_2 \in E(T)$ such that $w_1, w_2 \notin V_{t_1} \cap V_{t_2}$ and w_1 is separated from w_2 by $V_{t_1} \cap V_{t_2}$ in G.*

Proof. Let $f = tt'$ be an edge of T. If we can find $w_1, w_2 \subseteq W$ as desired, there is nothing more to prove. If not, one of the sets $U_t, U_{t'}$ contains W. We denote this set by U_f and look at the next edge. Provided that we have not found w_1, w_2, t_1, t_2 as described above after examining all edges of T, there has to be some $\bar{t} \in V(T)$ such that

$$V_{\bar{t}} = \bigcap_{e \in E(T)} U_e$$

holds, since T is connected and acyclic. Furthermore, we have $W \subseteq V_{\bar{t}}$. $\square$

Lemma 3.2.7 provides us with some useful information about the tree-decomposition of a graph G. It follows that the tree-width of G is affected by the cliques contained in G.

Lemma 3.2.8 *The vertex set of any complete subgraph of G is contained in some part of $(T, \mathcal{V})$.*

Proof. Let K be a complete subgraph of G. Since the vertices of K are pairwise adjacent, Lemma 3.2.7 ensures the existence of a part V_t in $\mathcal{V}$ containing $V(K)$. $\square$

Corollary 3.2.9 *For the clique number $\omega(G)$ of G we have*

$$\omega(G) - 1 \leq tw(G).$$

3.3 Other representations of tree-decomposition

An alternative representation can sometimes be more suitable to study a given problem. There exist several notions which are closely related to tree-decomposition. We present clique-sums, partial k-trees and chordal graphs and how they are linked to tree-decomposition.

3.3.1 Clique-sums

We have already seen how a tree-decomposition $(T, \mathcal{V})$ of a graph G gives us information about the separation properties of G. For an edge $tt' \in V(T)$, we know that $X = V_t \cap V_{t'}$ separates the two vertex sets U_t and $U_{t'}$ in G. We could also depict G as the result of "gluing" together the two subgraphs induced by U_t and $U_{t'}$ at X. Therefore the entire graph is composed of such subgraphs and a tree-decomposition provides us with "gluing instructions" for these parts. We state this precisely by introducing clique-sums.

Definition 3.3.1 Let $G = (V, E)$ be a graph. A set $C \subseteq V$ of k mutually adjacent vertices is called a *k-clique* in G.

Definition 3.3.2 Let G_1, G_2 be graphs and C_1 and C_2 be k-cliques in G_1 and G_2, respectively. The graph G is a *k-clique-sum* of G_1 and G_2 if G is obtained from G_1 and G_2 by identifying the vertices of C_1 with the vertices of C_2 and possibly deleting some edges between these k vertices. We also say that G is obtained by *pasting* the graphs G_1 and G_2 *together along* C_1 and C_2, or that G is a *clique-sum of order k*.

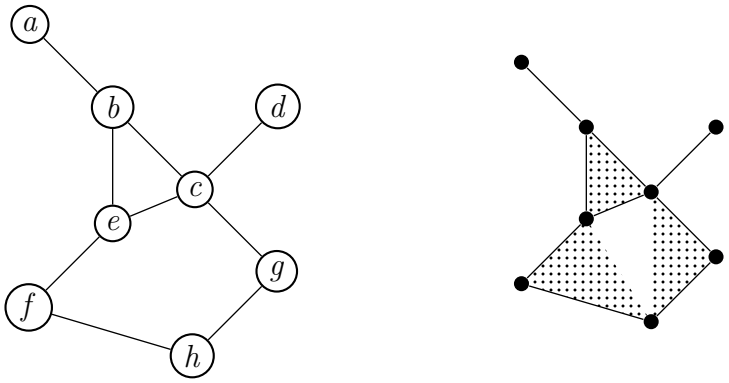

Figure 3.3: Graph of tree-width 2 and a pasting structure for clique-sums of graphs of cardinality ≤ 3 given by the tree-decomposition in fig. 3.1

Lemma 3.3.3 *If the graphs G_1 and G_2 have both tree-width at most n then every graph obtained by a clique-sum of G_1 and G_2 of order at most n also has tree-width at most n.*

Proof. Let C_1, C_2 be k-cliques of G_1 and G_2 for an integer $k \leq n$ and let G be the k-clique-sum of G_1 and G_2 obtained without deleting any edge. If $(T_1, \mathcal{V}_1)$ and $(T_2, \mathcal{V}_2)$ are tree-decompositions of G_1 and G_2, respectively, we know from Lemma 3.2.8 that there exist parts $V_1 \in \mathcal{V}_1$ and $V_2 \in \mathcal{V}_2$ such that $C_1 \subseteq V_1$ and $C_2 \subseteq V_2$. We now combine the two

tree-decompositions to a tree-decomposition of G by adding an edge e between the parts V_1 and V_2 and relabeling appropriately. The tree corresponding to this tree-decomposition consists of T_1, T_2 and e. □

Lemma 3.3.4 *Let G be a graph of tree-width n. Then G can be obtained by pasting together graphs of cardinality at most $n+1$.*

Proof. The pasting structure and graphs needed for this construction are given by any tree-decomposition of G of width n. □

3.3.2 Partial k-trees

Partial k-trees are an equivalent notion to graphs of tree-width at most k. Rose studied k-trees in the context of perfect elimination graphs and their connection to sparse linear systems [49, 50]. Arnborg et al. examined partial k-trees for their interesting property that many combinatorial problems can be solved in polynomial time when restricted to partial k-trees [2, 4]. Bodlaender covers this topic in surveys such as [12] and [11]. For the sake of simplicity, we will assume all graphs to be simple graphs for the rest of this section.

Definition 3.3.5 We define *k-trees* recursively:
(1) The complete graph K_k is a k-tree.
(2) Given a k-tree G, we can construct a k-tree by adding a new vertex v and edges from v to k mutually adjacent vertices of G.

Remark 3.3.6 *Note that by this definition every vertex in a k-tree G has degree at least $k-1$. If $V(G) \geq k+1$ then at least one vertex has exactly degree k. A k-tree with $k+1$ vertices is the complete graph K_{k+1}.*

Definition 3.3.7 A *partial k-tree* is a subgraph of a k-tree.

Rose showed the following characterisation of k-trees in [50].

Theorem 3.3.8 *A graph G is a k-tree if and only if it has the following three properties:*
(1) G is connected.
(2) G contains a k-clique but no $k+2$-clique.
(3) For every two vertices $v, w \in V(G)$ every minimal vw-separator in G is a k-clique.

Corollary 3.3.9 *For a k-tree G and for every two vertices $v, w \in V(G)$ every minimal vw-separator in G is also a minimal separator of G.*

Proof. Choose $v, w \in V(G)$ arbitrarily. Let S_{vw} be a minimal vw-separator in G. From Theorem 3.3.8 we know that S_{vw} is a k-clique. Suppose that S_{vw} is not a minimal separator of G. Then there exists a separator S of G of cardinality $|S| < |S_{vw}| = k$. By definition S separates some vertices v_1, v_2 in G. By Theorem 3.3.8 we have a contradiction as S cannot be both a k-clique and of order at most $k-1$. □

Definition 3.3.10 A vertex v of a graph G is *simplicial* if its neighbourhood $N_G(v)$ induces a complete subgraph in G.

Definition 3.3.11 Let $G = (V, E)$ be a graph with $n = |V|$. An *elimination ordering* $(v_i)_{i=1}^n$ is an ordering of the vertices in V. Set $G_0 = G$. *Applying* $(v_i)_{i=1}^n$ to G is the procedure of

(1) removing the vertex v_i from G_{i-1} and
(2) adding edges between non-adjacent vertices in $N_{G_{i-1}}(v_i)$ until they induce a complete subgraph in the now remaining graph G_i.

And this for i from 1 to n, in the order given by the elimination ordering.

Definition 3.3.12 A graph $G = (V, E)$ is a *perfect elimination graph* if it has an elimination ordering such that every vertex $v \in V$ is simplicial at the time of its removal. That is, if v is the ith vertex to be removed, v is simplicial in the graph

$$G[V \setminus \{v_1, \ldots, v_{i-1}\}].$$

Such an ordering is called *perfect elimination ordering*. Note that no edges are added to G in the process of applying a perfect elimination ordering.

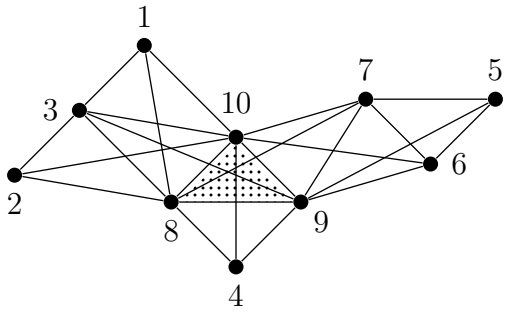

Figure 3.4: 3-tree with labels indicating a perfect elimination ordering

We can deduce from the recursive definition of k-trees that every k-tree has a perfect elimination ordering. Therefore the k-trees form a subclass of the class of perfect elimination graphs. Perfect elimination graphs are exactly the chordal graphs which we will study in the next section.

Definition 3.3.13 Define G as before. The *width* of an elimination ordering $(v_i)_{i=1}^n$ is

$$\max_{i \in \{1,\ldots,n\}} |N_i|$$

where N_i denotes the set of neighbours $N_{G_{i-1}}(v_i)$ of v_i at the time of its removal.

Theorem 3.3.14 *A graph G is a partial k-tree if and only if it has an elimination ordering of width at most k.*

Proof. Let G be a partial k-tree. Without loss of generality we can assume that G is a k-tree. The perfect elimination ordering implicitly given by the recursive definition of k-trees is of width at most k.

Conversely, let $G = (V, E)$ have an elimination ordering of width at most k. Assume that it is perfect. Therefore we can construct G recursively, starting with v_n and adding the other vertices in inverse order to our perfect elimination ordering. When we add a vertex v_i, we also add edges to the vertices in N_i which form a $|N_i|$-clique with $|N_i| \leq k$. From this construction we can easily deduce that G is a partial k-tree. □

The following theorem can be found for example in [11] or [55].

Theorem 3.3.15 *A graph $G = (V, E)$ is a partial k-tree if and only if G has tree-width at most k.*

Proof. Let G be a partial k-tree. Assume that G is even a k-tree. We apply induction on the number of vertices of G. For $|V| \leq k + 1$ it is obvious that G has tree-width at most k. Now let $|V| > k + 1$. There exists at least one vertex $v \in V$ of degree k whose neighbourhood $N(v)$ forms a k-clique in G. By induction hypothesis, there exists a tree-decomposition $(T, \mathcal{V})$ of $G - v$ of width at most k. Lemma 3.2.8 assures the existence of a part $V_t \in \mathcal{V}$ such that $N(v) \subseteq V_t$. Expanding $(T, \mathcal{V})$ by a new part

$$V_{t'} = N(v) \cup v$$

and adding an edge between t and t' in T results in a tree-decomposition of G of width at most k.

Conversely, let $(T, \mathcal{V})$ be a tree-decomposition of G of width at most k. Again, we apply induction on the number of vertices in V. If $|V| \leq k + 1$ then G is definitely a partial k-tree. If $|V| > k + 1$ we examine a leaf t of T. Let t' be the neighbour of t in T. If $V_t \subseteq V_{t'}$ then we delete V_t from $(T, \mathcal{V})$ and work with the resulting tree-decomposition of G of width at most k. So, let there be a vertex $v \in V_t \setminus V_{t'}$. This vertex v is not contained in any other part than V_t and is therefore adjacent to at most $|V_t| - 1 \leq k$ vertices in G. By induction hypothesis, $G - v$ is a partial k-tree. Adding v and at most k edges preserves this property for G as well. □

Corollary 3.3.16 *A graph G has tree-width at most k if and only if it has an elimination ordering of width at most k.*

Proof. Corollary 3.3.16 follows from Theorem 3.3.14 and Theorem 3.3.15. □

Lemma 3.3.17 *Every graph G of tree-width at most k contains a vertex of degree at most k.*

Proof. Suppose not. That is, all vertices have degree at least $k + 1$. This contradicts the fact that, by Theorem 3.3.15, G is a partial k-tree and as such can be constructed recursively by adding vertices of degree at most k. □

Arnborg et al. studied the complexity of deciding whether a given graph is a partial k-tree or not. They published the following result in [2].

Theorem 3.3.18 *Given an arbitrary graph $G = (V, E)$ and an arbitrary integer k, the problem of deciding if G is a partial k-tree is $\mathcal{NP}$-complete.*

If k is not part of the input but fixed then there exists an algorithm solving the aforementioned problem in time polynomial in $|V|$.

This applies for the complexity of testing for tree-width k as well. We will readdress this problem in a later chapter.

3.3.3 Chordal graphs

Chordal graphs have interesting properties with respect to their tree-decomposition as they exhibit a certain regular structure. They belong to the well-studied class of perfect graphs. Perfect elimination graphs and hence also k-trees are chordal graphs.

The following results about chordal graphs can be found in [16, 49, 22, 12, 11].

Definition 3.3.19 A graph G is *chordal* if every cycle C in G of length at least four contains a *chord*, that is, an edge between two vertices of C not adjacent in C.

Every induced subgraph $G[W]$, $W \subseteq V$, of a chordal graph G that forms a cycle is therefore a triangle. We call this an *induced cycle* of G. Chordal graphs are also called *triangulated* graphs.

Lemma 3.3.20 *Let $G = (V, E)$ be a chordal graph and let $W \subseteq V$ be a set of vertices in G. Then the induced subgraph $G[W]$ is chordal as well.*

Proof. Lemma 3.3.20 directly follows from the definition of chordal graphs. Clearly, every induced cycle in $G[W]$ is an induced cycle in G as well and therefore a triangle. □

Proposition 3.3.21 *Complete graphs are chordal.*

Proof. Since every set of three vertices of a complete graph G induces a triangle in G, G obviously is a chordal graph. □

Lemma 3.3.22 *A graph is chordal if and only if it can be constructed recursively by taking clique-sums, starting from complete graphs and without deleting edges in the process.*

Proof. Let G_1 and G_2 be two complete graphs which are pasted together along some complete subgraphs $C_1 \subseteq G_1$ and $C_2 \subseteq G_2$. When no edges are deleted to form the resulting graph G then every induced cycle of G has to be contained in either G_1 or G_2. Since G_1 and G_2 are chordal every such induced cycle is a triangle. We conclude that G is chordal. Moreover, every graph recursively constructed as described above is chordal as well.

Conversely, let $G = (V, E)$ be a chordal graph. We apply induction on the number of vertices $|V|$. If G is a complete graph then there is nothing more to show. If not, let $u, v \in V$ be two non-adjacent vertices of G. Let $S \subseteq V \setminus \{u, v\}$ be a minimal uv-separator in G. Denote by C_u the component of $G - S$ containing u. Set

$$G_1 = G[V(C_u) \cup S]$$

and

$$G_2 = G - V(C_u).$$

Then G is a clique-sum of G_1 and G_2 along S. By induction hypothesis, G_1 and G_2 are constructed as stated above. We now have to show that S is complete. Suppose not. Then there exist two vertices $x, y \in S$ which are not adjacent. Since S is minimal each of them has a neighbour in G_1 and there exists an xy-path P_1 with internal vertices in G_1. Likewise, there exists an xy-path P_2 with internal vertices in G_2. $P_1 \cup P_2$ forms a cycle of length at least four in G, without a chord between x and y. This is a contradiction to G being chordal. □

Theorem 3.3.23 *A graph $G = (V, E)$ is chordal if and only if there exists a tree-decomposition of G into complete parts, that is, every part induces a complete subgraph in G.*

Proof. Let G be a chordal graph. We apply induction on the number of vertices in G. If G is complete then a tree-decomposition of G consisting of only one part $V(G)$ has the desired property. Suppose that G is not complete. From Lemma 3.3.22 it follows that G can be seen as a clique-sum $G_1 \cup G_2$ of two graphs with $G_1 \cap G_2$ being a complete subgraph of G. By induction hypothesis, G_1 and G_2 both have a tree-decomposition into complete parts. Denote them by $(T_i, \mathcal{V}_i)$, for $i = 1, 2$. Since $G_1 \cap G_2$ is complete there have to exist parts $V_{t_1} \in \mathcal{V}_1$ and $V_{t_2} \in \mathcal{V}_2$, each containing $V(G_1 \cap G_2)$. Set T to be the union of the two trees T_1 and T_2 together with an edge t_1t_2 between t_1 and t_2. Then $(T, \mathcal{V}_1 \cup \mathcal{V}_2)$ forms a tree-decomposition of G into complete parts.

We prove the other direction with induction on $|V|$ as well. Let $(T, \mathcal{V})$ be a tree-decomposition of G into complete parts such that $|T|$ is minimal. If it consists of only one part then G is complete and hence chordal. Otherwise, let t_1t_2 be an edge of T. Using the same notation as in Lemma 3.2.6, we set $G_1 = G[U_1]$ and $G_2 = G[U_2]$. Both G_1 and G_2 have smaller order than G since otherwise $|T|$ would not be minimal. Therefore it follows from induction hypothesis that G_1 and G_2 are chordal. According to Lemma 3.3.22 both parts V_{t_1} and V_{t_2} induce complete subgraphs in G, hence $G_1 \cap G_2$ is complete. So G can be obtained by a clique-sum of G_1 and G_2 along $G_1 \cap G_2 \subseteq G$. From Lemma 3.3.22 it follows that G is chordal. □

Theorem 3.3.23 gives us information about the tree-width of a chordal graph. We already know from Corollary 3.2.9 that for every graph G

$$\omega(G) - 1 \leq tw(G)$$

holds. Here, $\omega(G)$ denotes the clique number of G. If a graph G is chordal then Theorem 3.3.23 assures the existence of a tree-decomposition of width at most $\omega(G) - 1$. Hence we deduce the following result.

Corollary 3.3.24 *Let G be a chordal graph. Then G has tree-width $\omega(G) - 1$.*

Corollary 3.3.25 *Let $G = (V, E)$ be a graph. Then we have*

$$tw(G) = \min\{\omega(H) - 1 \mid G \subseteq H, H \text{ chordal}\}.$$

Proof. By Corollary 3.3.24, every chordal supergraph H of G has tree-width $\omega(H) - 1$. And from Lemma 3.1.6 we know that since G is a subgraph of H its tree-width $tw(G)$ is bounded from above by $\omega(H) - 1$.

Conversely, let $(T, \mathcal{V})$ be a tree-decomposition of G of width $tw(G)$. Consider the graph $H = (V, E')$, consisting of the vertex set V and an edge $uv \in E'$ whenever there exists a part $V_t \in \mathcal{V}$ such that $u, v \in V_t$. By (TD2), the graph H is a supergraph of G. Furthermore, $(T, \mathcal{V})$ is a tree-decomposition of H of width $tw(G)$ where every part induces a complete subgraph in H. From Theorem 3.3.23 we can deduce that H is chordal and its tree-width $\omega(H) - 1$ is at most $tw(G)$. □

In the proof of Corollary 3.3.25 we constructed a chordal supergraph H of a graph G by adding edges until every part of a given tree-decomposition $(T, \mathcal{V})$ of G induced a complete subgraph in H. In our representation of $(T, \mathcal{V})$ as a union of subtrees this would mean that for every two vertices in H their corresponding subtrees intersect if and only if there exists a part of $(T, \mathcal{V})$ containing both vertices.

Definition 3.3.26 Let $\mathcal{S}$ be a finite family of non-empty sets. The *intersection graph* of $\mathcal{S}$ is the graph with vertex set $\mathcal{S}$ and an edge between two vertices S_1 and S_2 if the two sets intersect.

Definition 3.3.27 A *subtree graph* is the intersection graph of a family of subtrees of a tree.

We see that an intersection graph corresponds to a tree-decomposition into complete parts. Gavril described this relation between chordal graphs and subtree graphs in [22].

Theorem 3.3.28 *A graph is a subtree graph if and only if it is a chordal graph.*

Rose proved the following characterisation of chordal graphs.

Theorem 3.3.29 *Given a graph G the following three statements are equivalent.*
(1) G is chordal.
(2) G is a perfect elimination graph.
(3) For every two vertices $v, w \in V(G)$ every minimal vw-separator in G is a complete subgraph of G.

The k-trees therefore form a subclass of the chordal graphs. We can even define a recursive constructing scheme for chordal graphs similar to our definition of k-trees.
(1) A complete graph is chordal.
(2) Given a chordal graph G, we can construct a chordal supergraph of G by adding a new vertex v and edges from v to all vertices of a complete subgraph H of G.
This recursive construction scheme is consistent with our previous results about chordal graphs and their particular structure regarding clique-sums and tree-decomposition.

3.4 Excluded minor theorems

We already know from Lemma 2.2.3 that we can describe every minor-closed class of finite graphs by means of excluded minors. For example, if we restrict the graphs considered to simple graphs, then the family of forests is minor-closed and consists of all graphs not containing the triangle K_3 as a minor. Equivalently, we could call it $Forb(K_3)$.

Conversely, it is also interesting to ask what excluding a certain graph H as a minor means for the structure of the graphs in $Forb(H)$. In the following we will present some excluded minor theorems. The case of excluding a planar graph plays an important part in the proof of the graph minor theorem and will be discussed in detail later on.

Theorem 2.2.8 tells us that excluding K_5 and $K_{3,3}$ as minors generates exactly the family of planar graphs. But what if we only forbid one of these graphs as a minor? Wagner stated the following two theorems concerning the structure of graphs in $Forb(K_5)$ and $Forb(K_{3,3})$ [56, 30].

Theorem 3.4.1 *A graph is in Forb(K_5) if and only if it can be obtained by clique-sums of order at most 3 of planar graphs and subgraphs of the Wagner graph V_8 (shown in figure 3.5).*

Remark 3.4.2 *Here, the complexity lies in the only if direction. The if direction is easily seen. We know that planar graphs do not contain a K_5 minor, and V_8 does not either. A clique-sum of order at most three of graphs not containing a K_5 minor cannot have a K_5 minor.*

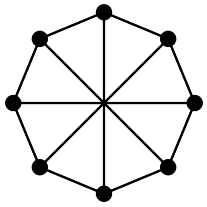

Figure 3.5: the Wagner graph V_8

Theorem 3.4.3 *A graph is in Forb($K_{3,3}$) if and only if it can be obtained by clique-sums of order at most 2 of planar graphs and copies of K_5.*

Fellows and Langston studied the structure of graphs not containing a cycle of length at least k [18].

Theorem 3.4.4 *Let $k \geq 3$ be fixed. Every graph $G = (V, E)$ not containing a cycle of length k as a minor has tree-width at most $k - 2$.*

Proof. Mark one arbitrarily chosen vertex $r \in V$ as the root for a depth-first spanning tree of G. This is a spanning tree T obtained by depth-first search starting from the root $r = r(T)$. We will construct a tree-decomposition of width at most $k - 2$ using T. Note that E consists only of edges used for T and of *back edges*. A back edge is an edge not in T joining a vertex $v \in V$ to a vertex $w \in V$ lying on the path T^{vr} between v and r in T.

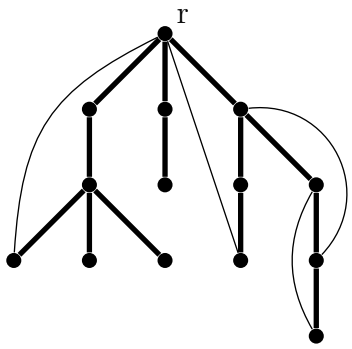

Figure 3.6: a graph G excluding a cycle of length 5 and a depth-first spanning tree T of G marked by thick edges; all unmarked edges are back edges

Recall the interpretation of T as a directed graph. Denote by $p(w)$ the predecessor of $w \in V \setminus \{r\}$ in T and set $p(r) = r$. For every $v \in V$, we define $p^0(v), p^1(v), \ldots, p^{k-2}(v)$

recursively by

$$p^0(v) = v$$
$$p^i(v) = p(p^{i-1}(v)) \quad \text{for } i = 1, \ldots, k-2.$$

Now we construct a tree-decomposition $(T, \mathcal{V})$ of G by setting

$$V_v = \{p^0(v), p^1(v), \ldots, p^{k-2}(v)\}$$

for every $v \in V = V(T)$. Property (TD1) is satisfied because every vertex $w \in V$ is contained in the part V_w. Since G contains no cycle of length k or more any back edge $vw \in E$ joins two vertices such that T^{vw} is of length at most $k-2$. Therefore $(T, \mathcal{V})$ has property (TD2) as well. Furthermore, the structure of $(T, \mathcal{V})$ implies (TD3). We have therefore found a tree-decomposition of G of width at most $k-2$. □

Chapter 4

Tree-Decomposition in Graph Minor Theory

Tree-decompositions are a useful tool for the proof of the graph minor theorem. One of the main parts of the proof deals with the case of excluding a planar graph as a minor. In this chapter, we will discuss this part in detail. For this, we introduce brambles and grids and some of their important properties regarding tree-width. See also [16, 5, 8]. Then we present simplifications of the proofs given by Robertson and Seymour in [42] and [43] which can be found in [17] and [23]. We also explain how the results of this chapter can be expanded to graphs embeddable in higher surfaces and we state a generalisation of Kuratowski's theorem.

Tree-decompositions are also used by Robertson and Seymour to describe the general structure of graphs excluding an arbitrary but fixed graph G as a minor. In this thesis, however, we will focus our attention on the special case that G is planar.

4.1 A weaker version of the graph minor theorem

In this chapter, we study the implications of excluding a planar graph as a minor. Robertson and Seymour showed in [42] that, given a planar graph H, all graphs in $Forb(H)$ have bounded tree-width. Together with the fact (which was proven in [43]) that graphs of bounded tree-width are well-quasi-ordered this implies the following weaker version of the graph minor theorem.

Theorem 4.1.1 *Let $G_0, G_1, G_2, \ldots$ be a countable sequence of graphs and let G_0 be planar. Then there exist indices $0 \leq i < j$ such that $G_i \lesssim G_j$.*

The proofs given in [42] and [43] are long and technical and since their publication they have been greatly simplified. Here, we present shorter proofs for which we need the notions of grids and of brambles.

4.2 Grids and brambles

Both grids and brambles allow to deduce certain information about the tree-width of a graph G due to the specific regular structure they exhibit. Large grid minors and large brambles pose obstructions to small tree-width. In the following, we use brambles to determine the tree-width of grids.

Definition 4.2.1 The *r-grid* is the graph with vertex set $V = \{1, \ldots, r\}^2$ and edge set

$$E = \{(i,j)(k,l) \mid |i-k| + |j-l| = 1\}.$$

Grids are obviously planar graphs. On the other hand, every planar graph G is a minor of a (sufficiently large) grid. This can be seen by constructing a graph $G^{(3)}$ from G by splitting up vertices such that every vertex of $G^{(3)}$ has degree ≤ 3. For $G^{(3)}$ we find a sufficiently fine grid containing $G^{(3)}$ as a topological minor; see [6].

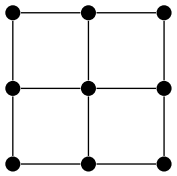

Figure 4.1: 3-grid

In general, it is not easy to determine the tree-width of a given graph $G = (V, E)$. But there are certain obstructions to small tree-width, one of them are large brambles. Bellenbaum and Diestel give some useful proofs concerning brambles [5].

Definition 4.2.2 Two sets $A, B \in V$ *touch* if they have a vertex in common or there exists an edge in G with endvertices in A and B. A *bramble* is a set $\mathcal{B} \subseteq \mathfrak{P}(V)$ of mutually touching connected vertex sets. A *cover* of a bramble $\mathcal{B}$ is a set $S \subseteq V$ such that every set in $\mathcal{B}$ contains at least one vertex of S. The cardinality of a vertex-minimal cover of $\mathcal{B}$ is called the *order* of the bramble $\mathcal{B}$.

Lemma 4.2.3 *Any set separating two covers of a bramble $\mathcal{B}$ is also a cover of $\mathcal{B}$.*

Proof. Every set $B \in \mathcal{B}$ is connected and contains a vertex of each cover, therefore also a path between them. So every set separating the two covers contains at least one vertex of B. □

Definition 4.2.4 G has *bramble number* $\beta(G) = n$ if n is maximal such that G contains a bramble of order n.

Like complete subgraphs, the brambles contained in a given graph affect its tree-width.

Theorem 4.2.5 *Let $G = (V, E)$ be a graph. Then we have*

$$tw(G) = \beta(G) - 1.$$

Proof. Let $(T, \mathcal{V})$ be a tree-decomposition of G. First, we show that every bramble $\mathcal{B}$ of G is covered by some part V_i and we thus have $tw(G) \geq \beta(G) - 1$.

Like in the proof of Lemma 3.2.7 we examine every edge tt' of T. If $V_t \cap V_{t'}$ covers $\mathcal{B}$ there is nothing more to show. If not, at least one of the sets U_t, $U_{t'}$ covers $\mathcal{B}$ because the sets in $\mathcal{B}$ are connected and touch; see Lemma 3.2.3 and Lemma 3.2.4. We delete tt' and from now on we examine only edges in the component of $T - tt'$ containing $\mathcal{B}$. We

continue with this procedure until we have found a covering set in the form of $V_t \cap V_{t'}$ or have no edges left to examine in the remaining component. In the last case we have only one part left, this part covers $\mathcal{B}$.

Now we assume that G has tree-width $\geq k$ but contains no bramble of order $> k$. For this we need the notion of a $\mathcal{B}$-*admissible* tree-decomposition for a given bramble $\mathcal{B}$ in G. This is a tree-decomposition of G such that no part of order $> k$ covers $\mathcal{B}$. If we find a $\mathcal{B}$-admissible tree-decomposition for every bramble $\mathcal{B}$ in G (including the empty bramble $\emptyset$) we have the desired contradiction and therefore $tw(G) \leq \beta(G) - 1$.

We apply induction on the number of sets contained in a bramble of G, starting with a bramble of maximal cardinality (this is bounded from above by $2^{|V|}$). Let $\mathcal{B}$ be a bramble and let there be, by induction, a $\mathcal{B}'$-admissible tree-decomposition for every bramble $\mathcal{B}'$ with more sets than $\mathcal{B}$. $\mathcal{B}$ has order $l \leq k$ and let X be a set of cardinality l covering $\mathcal{B}$. It now suffices to find, for every component C of $G - X$, a $\mathcal{B}$-admissible tree-decomposition of $G[V(C) \cup X]$ with X as a part. By identifying the X-parts these tree-compositions form a $\mathcal{B}$-admissible tree-decomposition of G.

So we examine $\mathcal{B}' = \mathcal{B} \cup V(C)$, where C is a given component of $G - X$. Let W denote the set $V(C) \cup X$. If $\mathcal{B}'$ is not a bramble then $V(C)$ does not touch all sets of $\mathcal{B}$ and $V(C) \cup N(C)$ and X are the parts of a $\mathcal{B}$-admissible tree-decomposition of $G[W]$.

If $\mathcal{B}'$ is a bramble it contains more sets than $\mathcal{B}$ since $V(C)$ is not covered by X. Thus there has to be a $\mathcal{B}'$-admissible tree-decomposition $(T, \mathcal{V})$ of G. If $(T, \mathcal{V})$ is not already $\mathcal{B}$-admissible we use it to form a tree-decomposition $(T, \mathcal{W})$ of $G[W]$ as desired. We know that there exists $s \in T$ such that V_s covers $\mathcal{B}$ but has cardinality $|V_s| > k \geq l$. Lemma 4.2.3 allows us to use Menger's theorem to assure the existence of l vertex-disjoint paths P_x, $x \in X$, between V_s and X. Since V_s does not cover $\mathcal{B}'$, it has to be part of $G - C$ and we can also choose the paths P_x, $x \in X$, to lie in $G - C$. We now select for every x in X a t_x such that $x \in V_{t_x}$ and define the parts

$$W_t = (V_t \cap W) \cup \{x \in X \mid t \in T^{t_x s}\}$$

for $(T, \mathcal{W})$. $(T, \mathcal{W})$ is a tree-decomposition of $G[W]$ with $|W_t| \leq |V_t|$ because for every $y \in W_t \setminus V_t \subseteq X$ we know that $y \in G - C$ and that, by Lemma 3.2.6, V_t separates V_s and V_{t_x} and therefore contains a vertex of the path $P_y \subseteq G - C$ that cannot be an element of W_t. But this also says that V_t separates the connected set $B \in \mathcal{B}$ with $y \in B$, hence covers B. From this it follows that no W_t can cover a set from $\mathcal{B}$ that V_t does not cover. Furthermore, every W_t with $|W_t| > k \geq l = |X|$ has to contain a vertex from C. Such a set covering $\mathcal{B}$, and therefore $\mathcal{B}'$, would be a contradiction to the $\mathcal{B}'$-admissible tree-decomposition $(T, \mathcal{V})$.

As we have $W_s = X$, we can form a $\mathcal{B}$-admissible tree-decomposition of G. □

We can use Theorem 4.2.5 to determine the tree-width of the r-grid. The set of *crosses*

$$C_{ij} = \{(i, k) \mid k = 1, \ldots, r\} \cup \{(l, j) \mid l = 1, \ldots, r\},$$

with $i, j \in \{1, \ldots, r\}$, forms a bramble of order r since at least r vertices are needed to cover all rows and columns. To construct a bramble of order $r + 1$, we use the crosses of the $(r - 1)$-grid and add the sets

$$\{(r, k) \mid k = 1, \ldots, r - 1\} \quad \text{and} \quad \{(l, r) \mid l = 1, \ldots, r\}.$$

On the other hand, the tree-width of the r-grid can be bounded from above by r. We show this by giving a path-decomposition of order r; see [8]. Its parts are the sets

$$V_{r(i-1)+j} = \{(i,k) \mid k = j, \ldots, r\} \cup \{(i+1,l) \mid l = 1, \ldots, j\}$$

for $i = 1, \ldots, r-1$ and $j = 1, \ldots, r$. This proves the following lemma.

Lemma 4.2.6 *The r-grid has tree-width r.* □

4.3 Excluding a planar graph

Lemma 4.2.6 tells us that large grids have large tree-width. As grids are planar, they cannot contain non-planar minors. From this it follows that, for any non-planar graph H, the set $Forb(H)$ contains graphs of arbitrarily large tree-width. We conclude that the following lemma holds.

Lemma 4.3.1 *Let H be a given graph. If the graphs in $Forb(H)$ have bounded tree-width then H has to be planar.* □

Interestingly, the converse also holds. It suffices to show that graphs not containing a certain grid have bounded tree-width since every planar graph is a minor of some grid. In [17], Diestel et al. give a proof of the following theorem that is not as long and technical as the initial proof given by Robertson and Seymour in their paper of the Graph Minors series [42].

Theorem 4.3.2 *For every integer r there is an integer $g(r)$ such that every graph of tree-width at least $g(r)$ has an r-grid minor.*

This chapter is dedicated to the proof of Theorem 4.3.2 which roughly runs as follows. We introduce k-meshes which partition a given graph into two subgraphs of relatively small intersection while maintaining some good connection properties between them. Then we show that every graph G of sufficiently large tree-width has to contain a certain k-mesh. Such a k-mesh implies the existence of some minor of G of particularly regular structure. In our case, we will use a k-mesh to find a complete subgraph or an r-grid minor in G. As every sufficiently large complete graph contains an r-grid minor this will complete our proof.

With Theorem 4.3.2 it is not difficult to prove the converse of Lemma 4.3.1.

Theorem 4.3.3 *Let H be a given graph. If H is planar then the graphs in $Forb(H)$ have bounded tree-width.*

Proof. Let r be an integer such that the r-grid contains H as a minor. Now let G be a graph in $Forb(H)$. Suppose that G has tree-width at least $g(r)$. Then it follows from Theorem 4.3.2 that G contains an r-grid minor. Hence G contains H as a minor as well, which is a contradiction. Therefore, all graphs in $Forb(H)$ have tree-width less than $g(r)$. □

4.3.1 Separations and k-meshes

Definition 4.3.4 For a graph G and a set $S \subseteq V(G)$ we define $\bar{S}$ to be the set

$$\bar{S} = G[V(S) \cup N(S)] - E(G[N(S)]).$$

Definition 4.3.5 We call a set $S \subseteq V$ *externally k-connected* in $G = (V, E)$ if $|S| \geq k$ and if for all sets $U, W \subseteq S$ with $|U| = |W| = l \leq k$ there exist l disjoint UW-paths in G that meet S only in their endvertices.

Definition 4.3.6 (separation) A pair (A, B) of subgraphs of G such that $A \cup B = G$ and $E(A) \cap E(B) = \emptyset$ is a *separation* of G. $|A \cap B|$ is called *order* of (A, B).

Definition 4.3.7 (premesh) A separation (A, B) such that $E(G[A \cap B]) \subseteq E(A)$ and such that there exists $T_A \subseteq A$ with the properties

(i) T_A is a tree of maximum degree $\Delta(T_A) \leq 3$ and
(ii) $\forall v \in V(A \cap B) : \ (v \in T_A) \wedge (d_{T_A}(v) \leq 2)$ and
(iii) $\exists v \in V(A \cap B) : \ d_{T_A}(v) = 1$

is called *premesh*.

Definition 4.3.8 (k-mesh) A *k-mesh* is a premesh (A, B) with the additional property that $V(A \cap B)$ is externally k-connected in B.

Lemma 4.3.9 *Let $G = (V, E)$ be a graph and let $h \geq k \geq 1$ be integers. If G has no k-mesh of order h then G has tree-width less than $h + k - 1$.*

Proof. Without loss of generality, see Lemma 3.1.7, we may assume that G is connected. We want to show that G has a tree-decomposition of width $< h + k - 1$. Suppose not. Then let $W \subsetneqq V$ be a set of maximal cardinality such that

(1) $G[W]$ has a tree-decomposition $(T, \mathcal{V})$ of width $< h + k - 1$
(2) for every component C of $G - W$
  (2.1) $N(C) \subseteq W$ is contained in some part V_t.
  (2.2) $(G - C, \bar{C})$ is a premesh of order at most h.

$W \neq \emptyset$ since every vertex of G could be chosen as singleton $W = \{v\}$.

Let C be a connected component of $G - W$. We now show that $N(C)$ has cardinality h. Suppose not. Then we can expand W by one vertex $v \in C$ that is a neighbour of a leaf of T_{G-C}. $W' = W \cup \{v\}$ has properties (1) and (2) since $(T, \mathcal{V})$ expanded by an additional part $N(C) \cup \{v\}$ is a tree-decomposition of $G[W']$ of width less than $h + k - 1$. Moreover, we have $v \in N(C') \subseteq N(C) \cup \{v\}$ for every component $C' \subseteq C$ of $G - W'$ because C is connected. This contradicts the maximality of W, hence we have $|N(C)| = h$ for all connected components of $G - W$.

As G contains no k-mesh of order h, $N(C)$ cannot be externally k-connected in C for any connected component C of $G - W$. So, there exist $Y, Z \subseteq N(C)$ with $|Y| = |Z| \leq k$ and only $l \leq |Y|$ disjoint YZ-paths with internal vertices in C. Menger's theorem now tells us that there is a set S of cardinality l that separates Y from Z. Set $W' = W \cup S$. Every connected component C' of $G - W'$ that touches S has neighbours in at most one of the sets $Y \setminus S$ and $Z \setminus S$. Assume that it does not touch $Z \setminus S$. Then we have

$$N(C') \subseteq (N(C) \setminus Z) \cup S$$

and therefore

$$|N(C')| \leq \underbrace{|N(C)|}_{=h} \underbrace{-|Z| + |S|}_{<0} < h.$$

Let P_s, $s \in S \cap N(C')$, denote those subpaths of the l YZ-paths that connect Z and $S \cap N(C')$. Then, together with

$$T_{G-W'} = T_{G-W} \cup \{P_s \mid s \in S \cap N(C')\},$$

$(G - W', \bar{W}')$ is a premesh of order at most h. If we set $\mathcal{V}' = \mathcal{V} \cup \{S \cup N(C)\}$, and enlarge T accordingly, we have found a tree-decomposition $(T', \mathcal{V}')$ of $G[W']$ of width

$$|N(C)| + |S| - 1 < h + k - 1.$$

Since $S \setminus N(C)$ contains at least one vertex (C is connected and touches Y and Z), W' contradicts the maximality of W.

We conclude that $W = V$ and that G has tree-width less than $h + k - 1$. □

4.3.2 Finding an r-grid minor

Lemma 4.3.10 *Every tree T of order at least $r(r-1)$ has an r-tuple $(v_1, \ldots, v_r)$ of vertices such that $T^{v_i v_{i+1}}$ contains no other vertex except v_i and v_{i+1} of this r-tuple, and this for $i = 1, \ldots, r-1$. We call this a good r-tuple of T.*

Proof. A set of r leaves has the desired property. If T has less than r leaves, it has at least $r(r-1) - (r-1) = (r-1)^2$ inner vertices connecting its leaves. Therefore, there exists a path of r vertices in T which can also be used to form a good r-tuple. □

We want to find a grid minor in a sufficiently large graph. One possibility is to construct it from a set of paths that already have a suitable structure.

Lemma 4.3.11 *Let $d, r \geq 2$ be integers such that $d \geq r^{2r+2}$. Let $G = (V, E)$ be a graph containing a set $\mathcal{H}$ of $r^2 - 1$ disjoint paths $H_1, \ldots, H_{r^2-1}$ and a set $\mathcal{V}$ of d disjoint paths $V_1, \ldots, V_d$ such that each of them meets every path in $\mathcal{H}$. In addition, assume that each path $H \in \mathcal{H}$ consists of d consecutive vertex-disjoint segments such that V_i meets H only in its ith segment, for every $i = 1, \ldots, d$. Then G has an r-grid minor.*

Proof. The paths of $\mathcal{H}$ and $\mathcal{V}$ intersect in such a way that we can use them to form an r-grid minor of G. We will denote the vertices of this grid minor by

$$(i, j)_G, \text{ for } i, j = 1, \ldots, r.$$

The crucial part is to find subpaths of paths from $\mathcal{H}$ and $\mathcal{V}$ that do not intersect with too many others. Then the rows of our grid will be provided by $\mathcal{H}$ and by using paths of $\mathcal{V}$ we will construct its columns.

To find suitable paths to form our rows and columns we define graphs G_i, one for each V_i, $i \in \{1, \ldots, d\}$, consisting of $\mathcal{H}$ as vertex set and of an edge between two paths H_j and H_k whenever V_i contains a subpath meeting $\mathcal{H}$ exactly in its endvertices from H_j and H_k. Every graph G_i has a spanning tree T_i of order $\geq r(r-1)$ and therefore with a good

r-tuple. As there do not exist more than $(r^2)^r$ different r-tuples of paths from $\mathcal{H}$, we know some r^2 of the initial $d \geq r^2(r^2)^r$ trees T_i to have a good r-tuple $(H^1, \ldots, H^r)$ in common. These trees $T_{i_1}, \ldots, T_{i_{r^2}}$ (ordered by ascending index) tell us which paths from $\mathcal{V}$ to use to form our columns. The edge between $(j,k)_G$ and $(j+1,k)_G$ from our grid will be obtained by suppressing the internal vertices of a path V_k^j. Let $i = (k-1)r + j$. We know that there exists a sequence of subpaths of V_i that connects H^j and H^{j+1} without meeting any $H \in \mathcal{H}$ in its internal vertices. V_k^j consists of these subpaths and, if necessary, of parts of some paths from $\mathcal{H} \setminus \{H^1, \ldots, H^r\}$ to connect them. All edges constructed in this fashion are disjoint since they are obtained from distinct, and therefore disjoint, paths from $\mathcal{V}$.

Now it suffices to contract all edges contained in the minimal subpath $H - k^j$ of H^j that contains the ith segment of H^j for all i with $i_{(k-1)r+1} \leq i \leq i_{kr}$, to create the vertex $(j,k)_G$ of our grid minor. Segments with an index greater than r^2 will not be needed. □

Lemma 4.3.12 *Let $G = (V, E)$ be a bipartite graph with partitioning sets A and B, $|A| = \alpha$, $|B| = \beta$, and let $\gamma \leq \alpha$ and $\delta \leq \beta$ be positive integers. If $|E| \leq \frac{(\alpha-\gamma)(\beta-\delta)}{\delta}$ then there exist vertex sets $C \subseteq A$ and $D \subseteq B$ such that $|C| = \gamma$, $|D| = \delta$ and the set $C \cup D$ is stable in G.*

Proof. Choose d vertices from B such that each vertex has at most $(\alpha - \gamma)/\delta$ neighbours in A (less than $\beta - \delta$ vertices can have more than $(\alpha - \gamma)/\delta$ neighbours). D touches at most $\alpha - \gamma$ vertices in A. Hence there exist at least γ vertices that can be used to form $C \subseteq A$ such that $C \cup D$ is independent in G. □

Lemma 4.3.13 *Let T be a tree of maximum degree $\Delta(T) \leq 3$ and W be a vertex set $W \subseteq V(T)$. For every integer $k \geq 2$ there exists a set of edges $F \subseteq E(T)$ such that all components of $T - F$ have between k and $2k - 1$ vertices in W, except for one possibly containing less than k vertices.*

Proof. We prove this lemma by induction on the number of vertices in W. Induction starts with a set W with $|W| \leq 2k - 1$, where we can use $F = \emptyset$. For $|W| \geq 2k$, we choose an edge $e \in E(T)$ and a component C of $T - e$ of minimal order such that $|C \cap W| \geq k$. This minimality, together with $\Delta(T) \leq 3$, implies $|C \cap W| \leq 2k - 1$. Our desired set F consists of e and the set F' that we know, by induction hypothesis, exists for $T - C$. □

Obviously, the r-grid is a minor of K_{r^2}. Therefore the following theorem implies Theorem 4.3.2.

Theorem 4.3.14 *Let r, m be positive integers and let $G = (V, E)$ be a graph of tree-with at least $r^{4m^2(r+2)}$. Then G contains either K_m or the r-grid as a minor.*

Proof. For the same reason as aforementioned we can assume that $2 \leq m \leq r^2$. The theorem obviously holds for $r = 1$, so from now on we will assume $r \geq 2$ as well.

By using the preceding lemmata we want to find a suitable k-mesh of G that will provide us with sufficiently many paths to form either a K_m minor or an r-grid minor.

For this, set $c = r^{4(r+2)}$ and $k = c^{m(m-1)}$. Then we have $c \geq 2^{16}$, $c^m \geq 2m+3$ and therefore

$$\begin{aligned} tw(G) \geq r^{4m^2(r+2)} = c^{m^2} = c^m k &\geq (2m+3)k - (m+2) \\ &= (m+1)(2k-1) + k - 1. \end{aligned}$$

Lemma 4.3.9 guarantees us the existence of a k-mesh (A, B) of order $(m+1)(2k-1)$. If T_A is a corresponding tree, we know that $V(A \cap B) \subseteq V(T_A)$, and we use Lemma 4.3.13 to partition T_A into

$$\frac{|V(A \cap B)|}{2k-1} - 1 = \frac{(m+1)(2k-1)}{(2k-1)} - 1 = m$$

disjoint subtrees $T_1, \ldots, T_m$, each with at least k vertices from $V(A \cap B)$. Set $A_i = V(T_i)$, for $i = 1, \ldots, m$. As $V(A \cap B)$ is externally k-connected in B there exists a set $\mathcal{P}_{ij}$ of k disjoint A_iA_j-paths that meet A only in their endpoints, and this for $1 \leq i < j \leq m$. By fixing an arbitrary bijection

$$\sigma : \{ij \mid 1 \leq i < j \leq m\} \to \{0, 1, 2, \ldots, \tbinom{m}{2} - 1\},$$

we can order these sets of paths. We will use them either to construct a K_m minor - where the sets A_i will play the part of vertices - or to find two sets $\mathcal{P}_{pq}$ and $\mathcal{P}_{ij}$ that satisfy the preconditions of Lemma 4.3.11.

For this purpose, let $l^* \leq \binom{m}{2}$ be the maximal integer such that for all $1 \leq i < j \leq m$ and all $0 \leq l < l^*$ there exist sets $\mathcal{P}^l_{ij}$ that satisfy the following five conditions that impose a suitable structure on the set of all these paths. The value of l^* will tell us which kind of minor we will be able to form.

(1) First of all, $\mathcal{P}^l_{ij} \neq \emptyset$ is a set of disjoint A_iA_j-paths such that only their endvertices lie in A.

Secondly, we set limits on the size of the sets $\mathcal{P}^l_{ij}$ in order to maintain our desired structure and to assure $|\mathcal{P}^l_{ij}| \geq c^2$ for $\sigma(ij) \geq l$.

(2) For $\sigma(ij) < l$, $\mathcal{P}^l_{ij}$ consists of precisely one A_iA_j-path that has no intersections with any path from $\bigcup_{st \neq ij} \mathcal{P}^l_{st}$.
(3) In the case of $\sigma(ij) = l$, $\mathcal{P}^l_{ij}$ contains exactly $\frac{k}{c^{2l}}$ paths.
(4) If $\sigma(ij) > l$, then we have $|\mathcal{P}^l_{ij}| = \frac{k}{c^{2l+1}}$.

Finally, we force the sets $H^l_{ij} = \bigcup \mathcal{P}^l_{ij}$ to coincide as much as possible. This last condition will later help us to construct an r-grid minor.

(5) For $l = \sigma(pq)$ and all $e \in E(H^l_{ij}) \setminus E(H^l_{pq})$ the graph $(H^l_{ij} \cup H^l_{pq}) - e$ does not contain $\frac{k}{c^{2l+1}}$ disjoint A_iA_j-paths.

We prove $l^* > 0$ by defining sets $\mathcal{P}^0_{ij}$, for all $1 \leq i < j \leq m$, that fulfil all five conditions. Set $\mathcal{P}^0_{pq} = \mathcal{P}_{pq}$ for $pq = \sigma^{-1}(0)$. If $\sigma(ij) > 0$, set H_{ij} to be the union of all paths in $\mathcal{P}_{ij}$ and let F_{ij} be a maximal set of edges from $H_{ij} \setminus \mathcal{P}^0_{pq}$ such that $(H_{ij} \cup \mathcal{P}^0_{pq}) - F_{ij}$ still contains a set of $\frac{k}{c}$ disjoint A_iA_j-paths - which we set to be $\mathcal{P}^0_{ij}$. Thus, we know that l^* is strictly greater than 0.

In the case of $l^* = \binom{m}{2}$, we can easily form a K_m minor by merging the sets A_1, $A_2, \ldots, A_m$ to vertices $v_1, v_2, \ldots v_m$ and by contracting the single path P_{ij} in $\mathcal{P}^{l^*-1}_{ij}$ to be the edge v_iv_j, for all pairs ij.

For $l^* < \binom{m}{2}$, we have to find two sets of paths with a suitable structure to utilise Lemma 4.3.11. Let $l = l^* - 1$ and $pq = \sigma^{-1}(l)$.

Suppose that there is a path $P \in \mathcal{P}^l_{pq}$ that avoids, for all ij with $\sigma(ij) > l$, a set $\mathcal{Q}_{ij}$ of some $\frac{1}{c}|\mathcal{P}^l_{ij}|$ of the paths in $\mathcal{P}^l_{ij}$. Then we could set

$$\begin{aligned}
\mathcal{P}^{l+1}_{ij} &:= \mathcal{P}^l_{ij}, \text{ for } \sigma(ij) < l,\\
\mathcal{P}^{l+1}_{pq} &:= P,\\
\mathcal{P}^{l+1}_{st} &:= \mathcal{Q}_{st}, \text{ for } \sigma(st) = l+1,
\end{aligned}$$

and for $\sigma(ij)$, like before, set F_{ij} to be a maximal set of edges from $E\left(\bigcup \mathcal{Q}_{ij}\right) \setminus E(H^{l+1}_{st})$ such that the graph

$$\left(\bigcup \mathcal{Q}_{ij} \cup H^{l+1}_{st}\right) - F_{ij}$$

still contains a set $\mathcal{P}^{l+1}_{ij}$ of $\frac{1}{c^2}|\mathcal{P}^l_{ij}|$ disjoint A_iA_j-paths.

As this would contradict the maximality of l^*, for every path $P \in \mathcal{P}^l_{pq}$ there exists a pair ij with $\sigma(ij) > l$ and P fails to meet at most $\frac{1}{c}|\mathcal{P}^l_{ij}| - 1$ paths of $\mathcal{P}^l_{ij}$. We can conclude that there is a set $\mathcal{P} \subseteq \mathcal{P}^l_{pq}$ of at least $\lceil |\mathcal{P}^l_{pq}| / \binom{m}{2} \rceil$ paths that have the same pair ij in common. We fix ij for the rest of the proof and have now the two sets of paths $\mathcal{P}$ and $\mathcal{P}^l_{ij}$ from where we will pick the sets $\mathcal{V}$ and $\mathcal{H}$ for Lemma 4.3.11. For this purpose, we define a bipartite graph G_B with partitioning sets $\mathcal{P}$ and $\mathcal{P}^l_{ij}$ and edge set

$$\{PQ \mid P \in \mathcal{P}, Q \in \mathcal{P}^l_{ij}, P \cap Q = \emptyset\}.$$

If we set $\gamma = \lceil \frac{1}{2}|\mathcal{P}| \rceil$ and $\delta = r^2$, we know that there exist sets $\mathcal{V}' \subseteq \mathcal{P}$ and $\mathcal{H} \subseteq \mathcal{P}^l_{ij}$ with $|\mathcal{V}'| = \gamma$ and $|\mathcal{H}| = \delta$ such that every path $V \in \mathcal{V}'$ meets all paths in $\mathcal{H}$. This follows from Lemma 4.3.12 and the fact that $d_{G_B}(P) < \frac{1}{c}|\mathcal{P}^l_{ij}|$ for all $P \in \mathcal{P}$ and that therefore

$$\begin{aligned}
E(G_B) &< |\mathcal{P}| \cdot \tfrac{1}{c}|\mathcal{P}^l_{ij}|\\
&\leq \left\lfloor \tfrac{1}{2}|\mathcal{P}| \right\rfloor \tfrac{1}{2r^2}|\mathcal{P}^l_{ij}|\\
&\leq \left\lfloor \tfrac{1}{2}|\mathcal{P}| \right\rfloor \left(\tfrac{1}{r^2}|\mathcal{P}^l_{ij}| - 1\right)\\
&= \frac{(|\mathcal{P}| - \lceil \frac{1}{2}|\mathcal{P}| \rceil)(|\mathcal{P}^l_{ij}| - r^2)}{r^2}
\end{aligned}$$

holds. Our last task is now to choose paths from $\mathcal{V}'$ that meet the paths of $\mathcal{H}$ in exactly the right way to form an r-grid minor.

We set $d = \lfloor \sqrt{c}/m \rfloor$ and start by partitioning one fixed path $Q \in \mathcal{H}$ into d segments $Q'_1, Q'_2, \ldots, Q'_d$, each meeting sufficiently many paths from $\mathcal{V}$. Thinking of Q as a path from A_i to A_j, we find a partitioning point between Q'_k and Q'_{k+1} in the first edge $e_k \in Q \setminus H^l_{pq}$ on Q such that the first component of $Q - e_k$ meets at least $kd|\mathcal{P}^l_{ij}|$ paths from $\mathcal{V}$. For $k = 1, \ldots, d-1$, denote this first component by Q_k and set $Q_d = Q$ and $Q_0 = \emptyset$. Since the paths in $\mathcal{V}$ are pairwise disjoint, every segment $Q'_k = Q_k - Q_{k-1}$, $k = 1, \ldots, d$, meets at least $d|\mathcal{P}^l_{ij}|$ paths from $\mathcal{V}$. This construction is possible, as we have

$$|\mathcal{V}| \geq \frac{1}{2}\mathcal{P} \geq \frac{1}{2} \cdot \frac{2|\mathcal{P}^l_{pq}|}{m(m-1)} \geq \frac{c}{m^2}|\mathcal{P}^l_{ij}| \geq d^2|\mathcal{P}^l_{ij}|$$

by condition (3) and (4).

In order to segment all paths in $\mathcal{H}$, we have to find a set S of partitioning vertices. For $k = 1, \ldots, d-1$, we know from condition (4) and (5) that $H^l_{pq} \cup H^l_{ij} - e_k$ contains at most $|\mathcal{P}^l_{ij}| - 1$ disjoint A_iA_j-paths. So we find a set S_k of $|\mathcal{P}^l_{ij}| - 1$ vertices such that there exists no A_iA_j-path in $H^l_{pq} \cup H^l_{ij} - e_k - S_k$. Since

$$S = \bigcup_{k \in \{1,\ldots,d-1\}} S_k$$

contains less than $d|\mathcal{P}^l_{ij}|$ vertices, every segment Q'_k of Q meets a path $V_k \in \mathcal{V}$ that avoids S.

Every path $P \in \mathcal{P} \setminus \{Q\}$ contains exactly one vertex v_k from each S_k. Set $P_0 = \emptyset$, $P_d = P$ and denote by P_k the initial segment of $P - v_k$. We partition P into segments

$$P'_k = P_k - P_{k-1},\ k = 1, \ldots, d.$$

Now consider $V_k \in \mathcal{V}$. V_k cannot meet P_{k-1} as this would result in an A_iA_j-path contained in $P_{k-1} \cup V_k \cup (Q - Q_{k-1})$ avoiding both e_{k-1} and S_{k-1}, contradicting our choice of S_{k-1}. Analogously, V_k cannot meet $P - P_k$. Therefore every path $V_k \in \{V_1, \ldots, V_d\}$ meets each path from $\mathcal{H} \setminus \{Q\}$ exactly in its kth segment and we can apply Lemma 4.3.11. □

The lower bound of $r^{4m^2(r+2)}$ used for the last theorem is by no means optimal but it suffices to show the correctness of Theorem 4.3.2. Robertson, Seymour and Thomas suspect that a bound of order $\mathcal{O}(r^2 \log r)$ might be closer to the right answer [48].

4.4 Well-quasi-ordering graphs of bounded tree-width

The main goal of this section is to prove that the graphs of bounded tree-width are well-quasi-ordered by the minor relation $\lesssim$. For this, we will introduce a new term called *branch-width* which, like tree-width, measures the connectivity of a graph. One of the papers discussing branch-width is [25]. In [44], Robertson and Seymour showed that branch-width and tree-width are closely related. More precisely, if $bw(G)$ denotes the branch-width of a graph G, the following proposition holds.

Proposition 4.4.1 *For any graph G we have*

$$bw(G) \leq tw(G) + 1 \leq \frac{3}{2}\, bw(G).$$

Therefore, if graphs of bounded branch-width are well-quasi-ordered the same holds for graphs of bounded tree-width. The latter was proved within the graph minor series by Robertson and Seymour [43]. We will, however, present a proof by Geelen et al. utilizing branch-width [23].

More generally, branch-width is defined not only for graphs, but for symmetric submodular functions. Therefore, we define branch-decompositions at first in general for symmetric submodular functions and then in the context of graph theory. To prove that for every integer n the family $\mathcal{G}_n$ of graphs of branch-width at most n is well-quasi-ordered

we take an approach similar to the technique introduced by Nash-Williams in [37]. For this, we utilise branch-decompositions of the graphs in $\mathcal{G}_n$ and additionally define a specific labeling of their edges establishing a useful order on subgraphs of graphs in $\mathcal{G}_n$. Finally, the results of this section are used to prove Theorem 4.1.1.

4.4.1 Symmetric submodular functions and branch-width

Definition 4.4.2 Let S be a finite set. A function φ defined on the set $\mathcal{P}(S)$ is *symmetric* if

$$\forall A \subseteq S : \ \varphi(A) = \varphi(S \setminus A),$$

and *submodular* if

$$\forall A, B \subseteq S : \ \varphi(A \cup B) + \varphi(A \cap B) \leq \varphi(A) + \varphi(B)$$

holds. For $A, B \subseteq S$ with $A \cap B = \emptyset$ we additionally define

$$\varphi(A, B) = \min\{\varphi(X) \mid A \subseteq X, X \cap B = \emptyset\}.$$

We call S *ground set* of φ and denote it by S_φ.

Definition 4.4.3 A tree T is *cubic*, or *binary*, if all vertices in T have a degree of either 1 or 3.

Definition 4.4.4 A *branch-decomposition* of a symmetric submodular function φ is a cubic tree T where the elements of S_φ are injectively identified with leaves of T. To simplify notation, we will assume $S_\varphi \subseteq V(T)$ in the following.

Remark 4.4.5 *T is also allowed to have unlabeled leaves, as they are easy to remove if a branch-decomposition without unlabeled leaves is needed.*

Definition 4.4.6 A subset S'_φ of S_φ is *displayed* by a subtree T' of T if $T' \cap S_\varphi = S'_\varphi$. The sets *displayed by an edge* e of T are the sets displayed by the components of $T - e$. The symmetric function φ assigns the same value to these two sets, we call this the *width* $\varphi(e)$ of e.

We denote by T_k the subgraph of T induced by the set of edges that have width at least k.

Definition 4.4.7 The maximum occurring width of an edge in T is the *width* of the branch-decomposition T.

The *branch-width* of a symmetric submodular function φ is the minimum integer n such that there exists a branch-decomposition of φ of width n.

Definition 4.4.8 Let $G = (V, E)$ be a graph and $E' \subseteq E$. The *connectivity function* γ_G assigns to E' the cardinality of $\Gamma_G(E')$, where $\Gamma_G(E')$ is the set of vertices incident with both an edge in E' and an edge in $E \setminus E'$.

The connectivity function is symmetric and submodular. We use it to define branch-width of graphs.

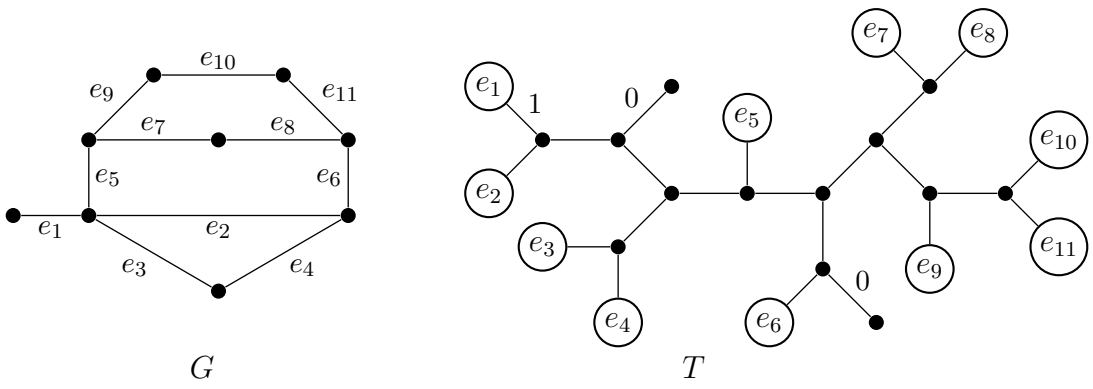

Figure 4.2: branch-decomposition T of width 2 of a series-parallel graph G, where all unlabeled edges have width 2

Definition 4.4.9 The *branch-width* of a graph $G = (V, E)$ is the branch-width of its connectivity function γ_G with ground set E. Analogously, we call the branch-decompositions of γ_G the *branch-decompositions* of G.

Lemma 4.4.10 *If H is a minor of a graph G with branch-width $bw(G)$ then H has branch-width at most $bw(G)$.*

Proof. We can assume that $|E(H)| \geq 2$, or else we have $bw(H) = 0$. Let T be the tree belonging to a branch-decomposition of G of minimal width and let T' be the smallest subtree of T such that $E(H) \subseteq T'$. It is possible that T' has inner vertices with degree 2. By taking T' and adding an edge with an unlabeled leaf to all of these vertices , we obtain a branch-decomposition of H with width at most $bw(G)$. □

Lemma 4.4.11 *For a connected graph $G = (V, E)$ the following holds:*

(i) $bw(G) = 0$ if and only if $|E| \leq 1$.

(ii) $bw(G) \leq 1$ if and only if there exists at most one vertex v with $d(v) \geq 2$.

Proof. (i) The first statement simply follows from our definition of branch-width.

(ii) The triangle K_3 and a path consisting of 4 vertices both have branch-width 2. Hence, from Lemma 4.4.10 we know that if $bw(G) \leq 1$ then G contains neither of these two graphs as a minor and this ensures our desired property. The converse obviously holds. □

It is easy to expand Lemma 4.4.11 to non-connected graphs simply by applying the given restrictions to all connected components.

Now consider two arbitrary different edges f and h of a tree T corresponding to a branch-decomposition of a symmetric submodular function φ. Denote by F the set displayed by the component of $T - f$ that does not contain h, and define H analogously. We call f and h *linked* if the minimum width over all edges in the edge-minimal path P in T containing f and h equals $\varphi(F, H)$. In any case, the width of every edge of P is at least $\varphi(F, H)$.

Definition 4.4.12 A branch-decomposition T is *linked* if every possible pair of edges of T is linked.

Definition 4.4.13 Let A and B be two sets. If neither $A \cap B$ nor $B \setminus A$ is empty, then A *splits* B.

Theorem 4.4.14 *Let φ be a symmetric submodular function with width n. Then φ has a linked branch-decomposition of width n.*

Proof. In order to find a linked branch-decomposition of φ, we define a partial order $\prec$ on the set of branch-decompositions of φ. For two branch-decompositions T and T' of φ, we say that $T' \prec T$ if there exists an integer k such that

$$|E(T'_k)| < |E(T_k)| \quad \text{or} \quad |E(T'_k)| = |E(T_k)| \ \wedge \ c(T'_k) > c(T_k)$$

and such that

$$\forall l > k : \ |E(T'_l)| = |E(T_l)| \ \wedge \ c(T'_l) = c(T_k).$$

As φ has branch-width n, all minimal elements in this partial order are branch-decompositions of width n. We choose one of them and denote it by T. Now, we want to show that this branch-decomposition T is linked. Suppose not. Then there exists a pair (f, h) of edges that are not linked. By forming a new branch-decomposition $\hat{T}$ of φ we will get the desired contradiction.

$\hat{T}$ is constructed as follows: Let $A \subseteq S_\varphi$ be a set containing F, but disjoint to H, such that $\varphi(A) = \varphi(F, H)$. In addition, A should split as few subsets of S_φ displayed by edges in T as possible. Set T^{uv} to be the (unique) path containing exactly one endvertex from each f and h, with $u \in f$ and $v \in h$. $\hat{T}$ consists of a copy T^+ of the component of $T - h$ containing f, a copy T^- of the component of $T - f$ containing h and an edge a joining the copies of u and v of degree 2. This duplicates exactly the component of $T - \{f, h\}$ containing u and v. If every element of S_φ is identified with its copy in T^+ if it is also an element of A, and with its copy in T^- otherwise, then $\hat{T}$ is a branch-decomposition of φ.

For every edge $e \in T$ and a copy $\hat{e}$ of e in $\hat{T}$ we have $\varphi(\hat{e}) \leq \varphi(e)$. To see this, let $\hat{e}$ be element of T^+ and denote by W the set displayed by the component of $T - e$ not containing v. Thus, we have $\varphi(e) = \varphi(W)$ and $\varphi(\hat{e}) = \varphi(W \cap A)$. The submodularity of φ together with the fact that $W \cup A$ contains F, but no vertex from H, and therefore has width $\varphi(W \cup A) \geq \varphi(F, H) = \varphi(A)$ gives us

$$\varphi(\hat{e}) + \varphi(W \cup A) \leq \varphi(e) + \varphi(A) \leq \varphi(e) + \varphi(W \cup A)$$

and thus $\varphi(\hat{e}) \leq \varphi(e)$.

Assume that equality holds here. Then we have $\varphi(W \cup A) = \varphi(A) = \varphi(F, H)$. Since A splits as few sets displayed by edges in T as possible, $W \cup A$ splits at least as many as A. Furthermore, W is displayed by e, so A contains either all or none of the vertices of W. It follows that A does not split W and one of the sets $W \setminus A$ and $W \cap A$ is empty. As we have

$$\varphi(A) = \frac{1}{2}\left(\varphi(A) + \varphi(S_\varphi \setminus A)\right) \geq \frac{1}{2}\left(\varphi(S_\varphi) + \varphi(\emptyset)\right) = \varphi(\emptyset)$$

for our symmetric and submodular function φ, we know that either $\varphi(W \setminus A) \leq \varphi(A)$ or $\varphi(W \cap A) \leq \varphi(A)$ holds. We want to examine the structure of $\hat{T}_{\varphi(A)+1}$. If there exists

another copy e^* of e in $\hat{T}$, it is of width

$$\varphi(e^*) = \begin{cases} \varphi(W \setminus A), & \text{if } e \notin T^{uv} \\ \varphi(W \cup A), & \text{if } e \in T^{uv}. \end{cases}$$

So, for $e \in E$ with $\varphi(e) = \varphi(\hat{e})$, and therefore $\varphi(W \cup A) = \varphi(A)$, we have either $\varphi(\hat{e}) \leq \varphi(A)$ or the width of its second copy is bounded above by $\varphi(A)$. Hence, $\hat{T}_{\varphi(A)+1}$ contains at most one copy of e.

This implies $|E(T_k)| \geq |E(\hat{T}_k)|$ for all $k \geq \varphi(A) + 1$. Moreover, we know for $k \geq \varphi(A) + 1$ that $\varphi(a) = \varphi(A)$ and therefore $c(\hat{T}_k) \geq c(T_k)$ in the case of $|E(T_k)| = |E(\hat{T}_k)|$. T is $\prec$-minimal by definition, hence we cannot have $\hat{T} \prec T$ and conclude that

$$\forall k \geq \varphi(A) + 1 : \ |E(T_k)| = |E(\hat{T}_k)| \ \wedge \ c(\hat{T}_k) = c(T_k)$$

holds. As a separates the component of $T_{\varphi(A)+1}$ containing $T^{uv} \cup \{f, h\}$ in $\hat{T}$, we have a contradiction. □

4.4.2 Well-quasi-ordering graphs of bounded branch-width

To show that the family $\mathcal{G}$ of graphs of bounded branch-width is well-quasi-ordered, we will use their branch-decompositions and a lemma on trees using a similar proof technique as for Kruskal's Theorem 2.2.9 on finite trees.

We have already defined rooted trees and how a rooted tree T can be interpreted as a directed graph, where each edge has a head and a tail and is directed away from the root $r(T)$. Now we additionally define rooted forests.

Definition 4.4.15 A *rooted forest* F contains countably many rooted trees. The set of the roots (leaves) of all these trees is the set of *roots (leaves)* of F. Edges incident with a root or a leaf are called *root edges*, or *leaf edges* respectively.

If $S \subseteq E(F)$ be a set of edges, then $u_F(S)$ denotes the set of edges whose tail is head of an edge in S.

A collection of countably many branch-decompositions can be interpreted as a binary forest (F, l, r).

Definition 4.4.16 A *binary forest* (F, l, r) consists of a rooted forest F containing only cubic trees such that every root has exactly one outgoing edge. Moreover, l and r are functions labeling for each nonleaf edge e its two succeeding edges by $l(e)$ and $r(e)$.

Definition 4.4.17 A map ψ assigning numbers from 0 to n to the edges of a graph G is called *n-edge labeling* of G.

If G is a rooted forest, an edge e of G *ψ-precedes* another edge f if $\psi(e) = \psi(f)$ and there exists a directed path P in G with first edge e and last edge f such that $\psi(g) \geq \psi(e)$ for every edge g on P. We also say that f *ψ-succeeds* e.

Lemma 4.4.18 *Let F be a rooted forest, ψ an n-edge labeling on F and $\preceq$ a quasi-order on $E(F)$ with no infinite strictly descending sequence and such that $e \preceq f$ if f ψ-precedes e. If $\preceq$ does not well-quasi-order $E(F)$, then there exists an infinite antichain A of edges such that $u_F(A)$ is well-quasi-ordered by $\preceq$.*

Proof. We prove this theorem by contradiction and assume that n is minimal such that we can find an n-edge labeling ψ of a forest F contradicting the theorem. Fix n, F and ψ for the rest of the proof. The set $N \subseteq E(F)$ of edges with label 0 is neither empty nor well-quasi-ordered since otherwise it would be possible to form an $(n-1)$-edge labeled counterexample by removing N and relabeling $E(F) \setminus N$. Like in the proof of Kruskal's theorem on trees, Theorem 2.2.9, we construct a bad sequence $a_0, a_1, a_2, \dots$ of edges from N such that

(1) for all $k \geq 0$ there exists a bad sequence starting with $a_0, a_1, \dots, a_{k-1}, a_k$, and
(2) there is no bad sequence starting with $a_0, a_1, \dots, a_{k-1}, e$ where e is an edge in $N \setminus \{a_k\}$ ψ-succeeding a_k.

Since infinite strictly descending chains are impossible, our sequence contains an infinite antichain A and by assumption, $u_F(A)$, and hence $u_F(\{a_0, a_1, \dots\})$, cannot be well-quasi-ordered. We form another counterexample with forest R that consists of all edge-maximal subtrees of F with root edges in $u_F(\{a_0, a_1, \dots\})$. R inherits this property from F because of the fact that $u_R(S) = u_F(S)$ for all $S \subseteq E(R)$. Hence, we find a bad sequence $b_0, b_1, b_2, \dots$ in $N \cap E(R)$. Every edge b_l of this sequence ψ-succeeds exactly one edge $a_{s(l)}$ from $a_0, a_1, \dots$. Let $s(k)$ be minimal such that there exists an edge b_k ψ-succeeding $a_{s(k)}$. Then the sequence

$$a_0, a_1, \dots, a_{s(k)-1}, b_k, b_{k+1}, \dots$$

is bad since a good pair (a_i, b_j) would imply $a_i \preceq b_j \preceq a_{s(j)}$ with $i < s(k) \leq s(j)$, which is a contradiction. □

Lemma 4.4.19 *Let F, ψ and $\preceq$ be given as in Lemma 4.4.18 with the only extension that (F, l, r) is an infinite binary forest. If the leaf edges of F are well-quasi-ordered by $\preceq$ but the set of root edges is not, then there exists an infinite antichain $e_0, e_1, e_2, \dots$ of nonleaf edges with the following two properties:*
(1) $l(e_0) \preceq l(e_1) \preceq l(e_2) \preceq \dots$
(2) $r(e_0) \preceq r(e_1) \preceq r(e_2) \preceq \dots$

Proof. Lemma 4.4.18 ensures us the existence of an antichain A such that the set $u_F(A)$ is well-quasi-ordered by $\preceq$. As an antichain, A can only contain finitely many leaf edges. Omitting them and repeatedly applying Lemma 2.1.10 to find suitable ascending sequences yields the desired result. □

We are now ready to prove the main theorem of this section. We will want to describe subgraphs from graphs and the vertices where they attach to their supergraphs. For this, we use the notion of rooted graphs.

Definition 4.4.20 A pair (G, R) is a *rooted graph* if G is a graph and R a subset of $V(G)$.

A *minor* of a rooted graph (G, R) is a rooted graph (G', R'). G' is a minor of G obtained by deletion of edges and vertices not in R, and by contracting edges. R' equals R, except that if an edge $uv \in E(G)$ containing a vertex from R' is contracted while constructing G', then R' changes to $(R' \setminus \{u, v\}) \cup \{v_{uv}\}$.

Lemma 4.4.21 *Let $G = (V, E)$ be a graph and $E_1 \subseteq E_2 \subseteq E$. Denote by G_1 and G_2 the subgraphs of G induced by E_1 and E_2 respectively. If*

$$\gamma_G(E_1) = \gamma_G(E_1, E \setminus E_2) = \gamma_G(E_2)$$

holds, then the rooted graph $(G_1, \Gamma_G(E_1))$ *is a minor of* $(G_2, \Gamma_G(E_2))$.

Proof. G_1 already is a minor of G_2. Menger's theorem guarantees the existence of $\gamma_G(E_1)$ disjoint paths in G_2 from $\Gamma_G(E_1)$ to $\Gamma_G(E_2)$. By contracting these paths, we obtain $(G_1, \Gamma(E_1))$ as a minor of $(G_2, \Gamma(E_2))$. □

Theorem 4.4.22 *The family* $\mathcal{G}_n$ *of graphs with branch-width at most* n *is well-quasi-ordered by the minor relation* $\lesssim$.

Proof. For every graph $G \in \mathcal{G}_n$, let T_G be a linked branch-decomposition of G of width at most n. Without loss of generality, we can assume that T_G contains an unlabeled leaf. If not, we simply subdivide an edge and add a leaf edge with an unlabeled leaf to the new vertex. The set F of all these linked branch-decompositions becomes a binary forest (F, l, r) by fixing l and r arbitrarily such that each nonleaf edge e is incident to exactly one left edge $l(e)$ and one right edge $r(e)$, both having the head of e as a tail.

We define a quasi-order $\preceq$ on $E(F)$ and an n-edge labeling ψ of F that fulfil the preconditions of Lemma 4.4.19. For every edge $e \in T_{G'}$ of some $G' \in \mathcal{G}_n$ we denote by E^e the set of edges displayed by the component of $T_{G'} - e$ not containing $r(T_{G'})$. G^e is the graph induced by E^e, and R^e the set $\Gamma_{G'}(E^e)$. Let f and h be edges from $E(F)$. Then $f \preceq h$ if the rooted graph (G^f, R^f) is a minor of the rooted graph (G^h, R^h). The function ψ defined by $\psi(e) = |R^e|$ for $e \in E(F)$ is an n-edge labeling of F and we have $e \preceq e'$ whenever an edge e' ψ-precedes another edge e.

$E(F)$ cannot contain an infinite strictly descending chain. Moreover, the root edges are not well-quasi-ordered by $\preceq$ as this would imply that the set $\mathcal{G}_n$ is well-quasi-ordered by the minor relation $\lesssim$. For every leaf edge $e \in E(F)$ the graph G^e contains at most one edge. Therefore, the set of leaf edges is well-quasi-ordered by $\preceq$ and we can apply Lemma 4.4.19 on (F, l, r), $\preceq$ and ψ. This gives us an infinite antichain $e_0, e_1, e_2, \ldots$ of nonleaf edges with properties (1) and (2). For every edge e in F, $(G^{l(e)}, G^{r(e)})$ is a separation of G^e of order $|R^{l(e)} \cap R^{r(e)}|$. As $R^{l(e)}$ and $R^{r(e)}$ both have cardinality at most n, we can find a subsequence E' of $e_0, e_1, e_2, \ldots$ such that all sets $R^{l(e')}$ with $e' \in E'$ have the same cardinality and the sets in $\{R^{r(e')} \mid e' \in E'\}$ have this property too. There are only finitely many combinations for mapping the vertices of a set $R^{l(e')}$ to the vertices in $R^{r(e')}$. Due to properties (1) and (2) of E', there exist edges e_i and e_j in E' such that $l(e_i) \preceq l(e_j)$, $r(e_j) \preceq r(e_j)$ and (G^{e_i}, R^{e_i}) is a minor of (G^{e_j}, R^{e_j}). As this means that $e_i \preceq e_j$ and E' is an antichain, we have a contradiction. Therefore, the set $\mathcal{G}_n$ is well-quasi-ordered. □

4.4.3 Implications of excluding a planar graph as a minor

We can deduce an analogous result to Theorem 4.4.22 for graphs of bounded tree-width.

Theorem 4.4.23 *The family of graphs with tree-width at most* n *is well-quasi-ordered by the minor relation* $\lesssim$.

Proof. This follows from Theorem 4.4.22 and Proposition 4.4.1. □

Together with the result from Theorem 4.3.3, that excluding a planar graph yields graphs of bounded tree-width, we can now prove Theorem 4.1.1.

Proof of Theorem 4.1.1. Suppose that the sequence

$$G_0, G_1, G_2, \ldots$$

is a bad sequence of graphs. Therefore there cannot exist any $j \geq 1$ such that $G_0 \lesssim G_j$. Hence all graphs G_j with $j \geq 1$ are in $Forb(G_0)$. Since G_0 is planar, it follows from Theorem 4.3.3 and Theorem 4.4.23 that the graphs in $Forb(G_0)$ are well-quasi-ordered by the minor relation $\lesssim$. The sequence

$$G_1, G_2, \ldots$$

is a countable sequence of graphs of $Forb(G_0)$, hence there have to exist indices $1 \leq i < j$ such that $G_i \lesssim G_j$. This is a contradiction to our assumption that the initial sequence is a bad sequence. We conclude that Theorem 4.1.1 is true. □

4.5 A generalisation of Kuratowski's theorem

From Theorem 2.2.8 we already know that the list of excluded minors is finite for the family of planar graphs. But what if we consider graphs embeddable in another surface than the plane? Theorem 4.4.23 and Theorem 4.3.2 can be used to derive a generalisation of Kuratowski's Theorem 2.2.7.

Theorem 4.5.1 *For every surface $\mathcal{S}$ the family $\mathcal{G}_\mathcal{S}$ of the graphs embeddable in $\mathcal{S}$ can be described by a finite list of excluded minors.*

The Graph Minor Theorem, Theorem 2.2.1, implies the above theorem. It is, however, possible to prove Theorem 4.5.1 directly. For this, we note that the family of graphs embeddable in a certain surface S is minor-closed. Furthermore, the graphs contained in the Kuratowski set $\mathcal{K}_{\mathcal{G}_\mathcal{S}}$ for $\mathcal{G}_\mathcal{S}$ cannot contain arbitrarily large grid minors [16]. Hence, it follows from Theorem 4.3.2 that the graphs in $\mathcal{K}_{\mathcal{G}_\mathcal{S}}$ have bounded tree-width and using Theorem 4.4.23 we conclude that Theorem 4.5.1 holds.

Chapter 5

Algorithmic Implications

The results published in the Graph Minors series of papers by Robertson and Seymour have a great impact on algorithmics concerning graphs, both from a theoretical and practical point of view. They proved the existence of a polynomial-time algorithm to decide if a given graph G contains a fixed graph H as a minor [45]. Together with the graph minor theorem, more precisely Corollary 2.2.4, this implies that membership in a minor-closed class of graphs can be decided in polynomial time, as there is only a finite number of minor testings to execute. This is, however, a purely theoretical bound on the time complexity of such an algorithm. Neither does it provide us with the actual list of excluded minors, nor does it say anything about the practicality of this algorithm. The list of excludes minors could be huge, as are the constants in the $\mathcal{O}(n^3)$ running time of the minor-testing algorithm of Robertson and Seymour. But it does give us information about the complexity of certain problems which can be proved to be minor-closed, even if a polynomial time algorithm has not yet been found.

In this chapter we will present some results concerning the computation of the tree-width of a given graph G [2, 9, 38].

We will further address improvements made in minor testing and examine a recent algorithm by Kawarabayashi et al. for the k disjoint paths problem. The main sources for this section were [45] and [29].

Then we show how the concept of tree-decomposition is used to make progress in solving ($\mathcal{NP}$-hard) problems in polynomial time on graphs of bounded tree-width. We give a brief overview of the topic and then present a generic approach for the construction of a polynomial time algorithm discussed by Bodlaender in [10].

Finally, we demonstrate the application of previously given results about brambles in the field of graph searching. A pursuit-evasion game discussed by Seymour and Thomas is presented [53].

5.1 Computing tree-width

In general, it is $\mathcal{NP}$-complete to compute the tree-width of a given graph G [2]. However, testing for a tree-width of at most a fixed integer k is less complex. Bodlaender proved the following theorem in [9].

Theorem 5.1.1 *Let k be a fixed integer. Then there exists an algorithm which*
given *a graph G*
outputs
either *a tree-decomposition of G of width k*
or *the result that $tw(G) > k$*
in $\mathcal{O}(|V(G)|)$ time.

Later, Perković and Reed improved Bodlaender's algorithm to give additional information in the case of $tw(G) > k$ [38]. This is useful for an improved algorithm solving the disjoint paths problem which we will discuss in the following section.

Theorem 5.1.2 *Let k be a fixed integer. Then there exists an algorithm which*
given *a graph G*
computes
either *a tree-decomposition of G of width k*
or *a subgraph G' of G of tree-width greater than k and a tree-decomposition of G' of width at most $2k$*
in $\mathcal{O}(|V(G)|)$ time.

It is desirable to have an algorithm that computes tree-width in an acceptable time as there are many problems that can be implemented efficiently for graphs of bounded tree-width. We will later present such problems.

5.2 Minor testing and the k disjoint paths problem

The question if a given graph G contains a fixed graph H as a minor arises naturally, not least due to the results obtained by N. Robertson and P. D. Seymour regarding the graph minor theorem. Within their framework of the Graph Minors series they introduced an algorithm to solve the k disjoint paths problem on a graph G in $\mathcal{O}(|V(G)|^3)$ time [45]. This fundamental problem is closely related to the problem of H minor containment. More precisely, Robertson and Seymour designed their algorithm to deal with δ-folios, a concept generalising the two aforementioned problems. Recently, in [29], Kawarabayashi, Kobayashi and Reed improved the complexity to $\mathcal{O}(|V(G)|^2)$ by optimizing the algorithm given in [45]. For planar graphs and graphs of bounded tree-width linear time algorithms exist, see [1, 39] for example.

Definition 5.2.1 (disjoint paths problem) Let $G = (V, E)$ be a graph. The *disjoint paths problem* deals with the question if, for given pairs $(s_1, t_1), (s_2, t_2), \ldots (s_k, t_k)$ of vertices of G, there exist k internally vertex-disjoint paths $P_1, P_2, \ldots, P_k$ in G, each P_i connecting s_i and t_i. If k is fixed, we also call it *k disjoint paths problem.* The vertices to be connected are named *terminals.*

For k not fixed, the disjoint path problem is $\mathcal{NP}$-complete, even for planar graphs [28, 29].

Definition 5.2.2 (disjoint connected subgraphs problem) Given a graph $G = (V, E)$ and non-empty subsets $Z_1, \ldots, Z_k \subseteq V$ with $\sum_{i=1}^{k} |Z_i| \leq t$, we want to know if there exist pairwise disjoint connected subgraphs $G_1, \ldots, G_k$ of G such that $Z_i \subseteq V(G_i)$ for $i \in \{1, \ldots, k\}$. Here, t is fixed.

To obtain the disjoint paths problem from the disjoint connected subgraphs problem, we only have to set Z_i to be the set $\{s_i, t_i\}$.

Definition 5.2.3 (H minor containment problem) Let H be a fixed graph. Given a graph G, we ask if G contains H as a minor.

Definition 5.2.4 Let (G, R) be a rooted graph, but now we order the vertices in R. We have

$$R = (v_1, v_2, \ldots, v_k) \in V(G)^k$$

as roots and write $(G, v_1, \ldots, v_k)$ for our graph (G, R). A graph $(H, u_1, \ldots, u_k)$ is a *minor* of $(G, v_1, \ldots, v_k)$ if there exists a mapping ϕ of H to G such that:

- Distinct edges of H are mapped to distinct edges of G.
- Every vertex $u \in V(H)$ is mapped to a non-empty connected subgraph $\phi(u)$ of G.
- For all $i \in \{1, \ldots, k\}$ the root v_i is contained in $V(\phi(u_i))$.
- For all $u \in V(H)$, $\phi(u)$ does not contain any edge that is an image of an edge of H under ϕ.
- For different vertices $u, w \in V(H)$ the subgraphs $\phi(u)$ and $\phi(w)$ are disjoint.
- If there exists an edge e joining u and w in H, then $\phi(e)$ has endvertices in $\phi(u)$ and in $\phi(w)$.

Definition 5.2.5 The set of all minors of a rooted graph $(G, v_1, \ldots, v_k)$ is its *folio*.

Definition 5.2.6 The *δ-folio* of a rooted graph $(G, v_1, \ldots, v_k)$ is a subset of the folio which contains all minors $(H, u_1, \ldots, u_k)$ such that
(1) $|E(H)| \leq \delta$ and
(2) $|V(H) - \{v_1, \ldots, v_k\}| \leq \delta$.

Definition 5.2.7 For a set $Z \subseteq V(G)$ of cardinality k the *δ-folio of G relative to Z* is the δ-folio of $(G, v_1, \ldots, v_k)$ for $v_1, \ldots, v_k$ arbitrarily chosen from $Z = \{v_1, \ldots, v_k\}$. The δ-folio of every other possible k-tuple of vertices from Z is thereby determined.

Definition 5.2.8 (folio problem) Given a graph $G = (V, E)$, $\delta \geq 0$ and $Z \subseteq V$, we want to determine the δ-folio of G relative to Z.

In [45], N. Robertson and P. D. Seymour give an algorithm solving the folio problem in $\mathcal{O}(|V|^3)$ time. To apply it to our problem of H minor containment, we simply have to set $Z = \emptyset$ and $\delta = \max\{|V(H)|, |E(H)|\}$. By running the 0-folio algorithm for G relative to $Z_1 \cup \cdots \cup Z_t$, we obtain a solution for the disjoint connected subgraphs problem described above. Therefore the folio problem also applies to the k disjoint paths problem.

Kawarabayashi et al. mostly stick to the structure of the algorithm given in [45], but restrict their paper to the case of solving the k disjoint paths problem. Algorithms for the other problems follow from adapting their proof to the δ-folio concept.

5.2.1 An $\mathcal{O}(|V(G)|^2)$ time algorithm for the k disjoint paths problem

In this section we will present the algorithm introduced by Kawarabayashi et al. in [29]. It is based on the algorithm and results given by Robertson and Seymour in [45] and improves the running time to $\mathcal{O}(|V(G)|^2)$.

The algorithm given by Robertson and Seymour for the k disjoint paths problem roughly runs as follows. Either the input graph G has bounded tree-width, then it is possible to solve the problem on G in linear time by dynamic programming. Or G has large tree-width. Then they search for a large t-clique minor that can be used to connect the terminals by disjoint paths. If there exists no such t-clique minor or if it can be separated from the terminals by a sufficiently small separation and is therefore not useful, they search for a vertex v that is *irrelevant* for the given problem. More precisely, the problem can be solved for G if and only if it can be solved in $G - v$. Removing such a vertex reduces the input graph for the next iteration step.

The proof of correctness of this algorithm needs numerous results from papers of the Graph Minors series. However, a shorter proof was developed just recently [31].

In this chapter we will sometimes denote the cardinality $|V(G)|$ of the input graph G by n. Analogously, we set m to be the number of edges.

5.2.1.1 Walls and t-certificates

Definition 5.2.9 An *elementary wall* of height r is a graph consisting of bricks, that is, cycles of length six, which are arranged in r rows of r bricks each. A *wall* of height r, or *r-wall*, is a subdivision of an elementary wall of height r. Obviously, walls are planar graphs and all vertices have degree at most 3. Four specific vertices are distinguished and called *corners*, see figure 5.1.

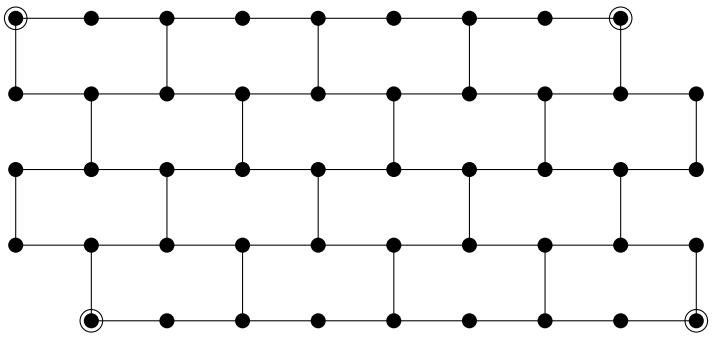

Figure 5.1: elementary wall of height 4 with marked corners

Remark 5.2.10 *Elementary walls have a structure very similar to the structure of grids. Elementary walls, however, have an advantage. Their vertices have a degree of at most three, in contrast to vertices of grids which mostly have degree four.*

Definition 5.2.11 The vertices of degree 3 in a wall W are called *nails* of W. Given a planar embedding of W, the *perimeter* $per(W)$ of W is the boundary of the unique face

in this embedding which contains $4(r-1)$ nails. If W is contained in a graph G, there exists a unique component C of $G - per(W)$ which contains $W - per(W)$. We call the induced subgraph $G[V(C) \cup V(per(W))]$ *compass* of W and denote it by $comp(W)$.

Definition 5.2.12 A *subwall* W' of a wall W is a subgraph of W which is a wall. If it consists of $h(W')$ consecutive bricks in $h(W')$ consecutive rows of W it is a *proper* subwall. It is *dividing* in G if $W - W'$ is disjoint from the compass of W' in G.

Walls are closely related to grids. Every graph G containing an r-wall as a minor also contains an $(r+1)$-grid minor. Conversely, if G contains an r-grid minor, it contains an $\lfloor r/2 \rfloor$-wall minor as well.

Theorem 5.2.13 *For every integer r there is an integer $f_1(r)$ such that every graph of tree-width at least $f_1(r)$ has an r-wall minor.*

Proof. This follows from Theorem 4.3.2 and the relation between grids and walls mentioned above. □

Theorem 5.2.14 *For every integer r and a graph G with $tw(G) \geq f_1(r)$, we can find an r-wall in G in linear time.*

Proof. We use the algorithm described in Theorem 5.1.2 to find a subgraph G' of G whose tree-width fulfils

$$f_1(r) \leq tw(G') \leq 2f_1(r).$$

From Theorem 5.2.13 follows that G' contains an r-wall minor which can be found in linear time by applying dynamic programming to the given tree-decomposition of G' of width at most $f_1(r)$. □

Definition 5.2.15 We call a wall W *flat* if there do not exist two vertex disjoint paths in $comp(W)$ joining diagonally opposite corners. This rather imprecise description refers to a representation of W as shown in figure 5.1.

Definition 5.2.16 Let W be a wall in a graph G. We say that $comp(W)$ can be *embedded into a plane up to 3-separations* if there exist *attached vertex sets*

$$A_1, A_2, \ldots, A_l \subseteq V(comp(W))$$

such that the following conditions hold.

(1) The sets $A_1, \ldots, A_l$ are pairwise disjoint and contain no corners of W.
(2) For all $i \in \{1, \ldots, l\}$ the neighbourhood $N(A_i)$ of A_i has cardinality $|N(A_i)| \leq 3$.
(3) For all $i, j \in \{1, \ldots, l\}$ with $i \neq j$ we have $N(A_i) \cap A_j = \emptyset$.
(4) Let W' denote the graph obtained from $comp(W)$ by deleting A_i and adding new edges joining every pair of distinct vertices in $N(A_i)$ for all $i \in \{1, \ldots, l\}$. Then W' can be drawn in a plane such that all corners of W are on the outer (unbounded) face boundary. We call this a *flat embedding*.

Seymour showed in [51] that flat walls can be characterised by such embeddings.

Theorem 5.2.17 *A wall W in G is flat if and only if $comp(W)$ can be embedded into a plane up to 3-separations.*

To improve the overall running time of the Robertson-Seymour-algorithm it is crucial to improve running times of subroutines that are of order $\mathcal{O}(m)$. As m is of order $\mathcal{O}(n^2)$, we try to bound the number of edges of the graph considered by $\mathcal{O}(n)$. Useful tools are t-certificates and a result by Nagamochi and Ibaraki.

Definition 5.2.18 Let $G = (V, E)$ be a graph. A *t-certificate* of G is a subgraph $G_t = (V, E_t)$ of G with the properties that
(1) $|E_t|$ is of order $\mathcal{O}(|V|)$ and
(2) for all $u, v \in V$ we have a vertex connectivity of $\kappa_{G_t}(u, v) \geq \min\{t, \kappa_G(u, v)\}$.

Note that G' is a spanning subgraph of G. Nagamochi and Ibaraki developed an algorithm which finds a t-certificate of G in $\mathcal{O}(|E|)$ time [36]. It will be the first step of the algorithm by Kawarabayashi et al.

Theorem 5.2.19 *There exists an algorithm to compute a t-certificate G' of a graph G in time $\mathcal{O}(m)$.*

5.2.1.2 Finding a large t-clique minor or an irrelevant vertex

Like the Robertson-Seymour-algorithm, our algorithm will use large t-clique minors to construct the desired paths or reduce the order of the graph considered. The question is whether we can find a large t-clique minor in G or not. The following theorem by Reed and Wood, published in [40], gives a sufficient condition for finding such a minor in linear time.

Theorem 5.2.20 *Given a graph $G = (V, E)$ with $|E| \geq 2^{l-3}|V|$ for some positive integer l, a K_l minor can be found in $\mathcal{O}(l(|V| + |E|))$ time.*

Remark 5.2.21 *To find a K_l minor using Theorem 5.2.20 it suffices to use only $2^{l-3}|V|$ edges of the graph G. Hence, in practice, we have an actual running time of order $\mathcal{O}(n)$.*

Given a sufficiently large wall in a graph G, we can also follow a different approach to find a t-clique minor.

Theorem 5.2.22 *Let $t, h \geq 2$ be fixed integers. Then there exists a computable constant $f_2(t, h)$ and an algorithm which*

given *a graph G and a wall W of height at least $f_2(t, h)$*

computes

either *a t-clique minor of G*

or *a subset $X \subseteq V(G)$ of order at most $\binom{t}{2}$ and t^2 disjoint proper subwalls*

$$W_1, W_2, \ldots, W_{t^2}$$

of height h. Each subwall is dividing and flat in $G - X$ and the algorithm returns a flat embedding for every subwall as well

in $\mathcal{O}(m)$ time.

Theorem 5.2.22 was already stated by Robertson and Seymour in [45], but with running time in $\mathcal{O}(nm)$. Kawarabayashi et al. improve this running time to $\mathcal{O}(m)$ by employing a different algorithm for the most expensive part. They use a test for flatness by Kapadia, Li and Reed which runs in $\mathcal{O}(m)$ time [27].

When we have found a sufficiently large t-clique minor, its usefulness depends on its position in the graph relative to the terminals. If it is not "too far away" from the terminals it will be possible to construct the desired paths. Robertson and Seymour used the following theorem.

Theorem 5.2.23 *Let $s_1, \ldots, s_k, t_1, \ldots, t_k$ be terminals and K a $|K|$-clique minor of order at least $3k$ in a graph G. If there does not exist a separation (A, B) of G of order at most $2k-1$ such that*

(1) A contains all terminals and
(2) $B - A$ contains at least one node of K

then k paths solving the k disjoint paths problem for the given terminals can be found in $\mathcal{O}(m)$ time.

By using t-certificates, Kawarabayashi et al. improve the running time to $\mathcal{O}(n)$.

Theorem 5.2.24 *Let k be a fixed integer.*

Given *a graph $G = (V, E)$ with terminals $s_1, \ldots, s_k, t_1, \ldots, t_k$*

and *a $3k$-clique minor K in G*

and *a $2k$-certificate $G_{2k} = (V, E_{2k})$ of G*

we can

either *find k disjoint paths $P_1, \ldots, P_k$, each P_i joining s_i and t_i, for $i = 1, \ldots, k$, in $\mathcal{O}(|V|)$ time,*

or *construct a graph $G' = (V', E')$ with $|V'| \leq |V| - 1$ and a $2k$-certificate $G'_{2k} = (V', E'_{2k})$ for G' such that V' contains all terminals and the k disjoint paths problem in G' is equivalent to the original problem. And this in $\mathcal{O}(|V| + |E \setminus E_{2k}|)$ time.*

Proof. We search for a separation fulfilling the properties 1 and 2 from Theorem 5.2.23. If the smallest such separation is of order at least $2k$, then we can utilise Theorem 5.2.23 to find k paths as desired.

Any such separation (A, B) of order at most $2k-1$ can be found by a simple flow algorithm applied to $K \cup G_{2k}$. Deleting $B - A$ from G and adding new edges joining every pair of distinct vertices in $A \cap B$ diminishes G to G' without changing the solvability of the given problem by Theorem 5.2.23. To obtain a $2k$-certificate of G', we execute the same procedure on G_{2k}. We then apply the algorithm of Theorem 5.2.19 in $\mathcal{O}(|V|)$ time since we have $|E_{2k}| + \binom{2k-1}{2}$ edges. □

From the last theorem it follows that, given a large t-clique minor in G, we can either solve the k disjoint paths problem or reduce it to a smaller instance by deleting vertices irrelevant for our problem. What if we do not have such a minor at our disposal? If G has bounded tree-width, Robertson and Seymour adapt a method introduced by Arnborg and Proskurowski in [4] to solve the δ-folio problem on G. Kawarabayashi et al. concentrate on the subproblem of finding *realisable partitions*, a problem corresponding to the disjoint connected subgraphs problem, and to 0-folio respectively.

Definition 5.2.25 Let $G = (V, E)$ be a graph and $Z \subseteq V$. A partition $\{Z_1, Z_2, \ldots, Z_p\}$ of Z is *realisable* if there exist disjoint trees $T_1, T_2, \ldots, T_p$ in G such that $Z_i \subseteq V(T_i)$ for $i = 1, \ldots, p$.

Furthermore, Kawarabayashi et al. modify the results used by Robertson and Seymour in that they assume the set of edges E to be of order $\mathcal{O}(|V|)$. This is due to the fact that their algorithm starts by repeatedly applying Theorem 5.2.20 and Theorem 5.2.24 to either solve the problem using a $3k$-clique minor or to reduce the number of edges in the graph considered. Therefore, the following theorems may state linear running times in n even if, in general, they are of order $\mathcal{O}(m)$.

Theorem 5.2.26 *Let w and k be fixed integers. Then there exists an algorithm which*

given *a graph $G = (V, E)$ with $|E|$ of order $\mathcal{O}(n)$ and a tree-decomposition of width at most w*

and *a set $Z \subseteq V$ of order at most $2k$*

enumerates all realisable partitions of Z in G. And this in $\mathcal{O}(f(k, w) \cdot n)$ time, for some function f of k, w.

Obviously, Theorem 5.2.26 solves the k disjoint paths problem for graphs of bounded tree-width. In the case that the input graph G has too large tree-width to apply Theorem 5.2.26 efficiently and no $3k$-clique minor can be found, the algorithm will search for an irrelevant vertex and remove it.

Kawarabayashi et al. extract the following theorem from [45] by combining several results and adapting them to the terminology used in their paper.

Theorem 5.2.27 *For a fixed integer k there exists a computable constant $f(k)$ such that*

given *a graph G and a subset $X \subseteq V(G)$ of order $|X| \leq \binom{3k}{2}$*

and *a flat wall W of height $f(k)$ in $G - X$*

there exists an irrelevant vertex $v \in V(W)$. Furthermore,

given *a flat embedding of $comp(W)$ with attached vertex sets $A_1, \ldots, A_l$*

and, *for $i = 1, \ldots, l$, all realisable partitions of Z_i in A'_i where*

$$A'_i = A_i \cup X \cup (N(A_i) \cap V(comp(W))) \text{ and}$$
$$Z_i = X \cup (N(A_i) \cap V(comp(W)))$$

an irrelevant vertex v can be found in linear time.

They further improve Theorem 5.2.27 by replacing the condition concerning realisable partitions by adding restrictions to the attached vertex sets $A_1, \ldots, A_l$.

Theorem 5.2.28 *For any fixed integer k there exist computable constants $f(k), g(k)$ and an algorithm which*

given *a graph G and a subset $X \subseteq V(G)$ of order $|X| \leq \binom{3k}{2}$*

and *a flat wall W of height $f(k)$ in $G - X$*

and *a flat embedding of $comp(W)$ with attached vertex sets $A_1, \ldots, A_l$ such that all the components $G[A_1], \ldots, G[A_l]$ have tree-width at most $g(k)$*

finds an irrelevant vertex v in $\mathcal{O}(n)$ time.

Proof. We want to utilise Theorem 5.2.27. For every attached vertex set A_i with $i \in \{1, \ldots, l\}$, the induced subgraph $G[A_i]$ of G has bounded tree-width. Thus the same holds for A'_i defined as in Theorem 5.2.27. So we can use Theorem 5.1.1 to find a tree-decomposition of $G[A'_i]$ in $\mathcal{O}(|A'_i|)$ time. Then, by Theorem 5.2.26, we can enumerate all realisable partitions of Z_i in A'_i, in $\mathcal{O}(|V|)$ time as well. From $|N(A_i) \cap V(comp(W))| \leq 3$ and $|X| \leq 9k^2$ it follows that

$$\sum_{i=1}^{l} |A'_i| \leq \sum_{i=1}^{l} |A_i| + (|X| + 3)l$$

which is in $\mathcal{O}(n)$. Thus, combined with Theorem 5.2.27, the algorithm runs in $\mathcal{O}(n)$ time. □

What is missing so far is the confirmation that we can find X and W as required by Theorem 5.2.28. Kawarabayashi et al. show that this is possible in linear time.

Theorem 5.2.29 *For any fixed integer k there exist computable constants $h(k)$ and $g(k)$ and an algorithm which*

given *a graph $G = (V, E)$ of tree-width at least $h(k)$*

computes

either *a $3k$-clique minor of G*

or *a pair (X, W) satisfying the conditions for Theorem 5.2.28.*

in $\mathcal{O}(n)$ time.

Proof. Set $h(k) \geq f_1(f_2(3k, f(k)))$ with $f(k)$ as in Theorem 5.2.28, $f_1(k)$ as in Theorem 5.2.13 and $f_2(k)$ as in Theorem 5.2.22. Furthermore, set $g(k) \geq h(k)$.

If G has at least $2^{3k-3}|V|$ edges, then, by Theorem 5.2.20, we can find a $3k$-clique minor.

If not, Theorem 5.2.14 tells us that we can find a wall W of height $f_2(3k, f(k))$ in G in linear time. Now Theorem 5.2.22 either gives us a $3k$-clique minor of G or a set $X \subseteq V$ of order $|X| \leq \binom{3k}{2}$ and $9k^2$ disjoint proper subwalls of height $f(k)$ as described in Theorem 5.2.22. We still have an overall running time linear in n, as $|E|$ is of order $\mathcal{O}(n)$.

In the case that we have not yet found a $3k$-clique minor, we choose two of the subwalls and denote them by W_1 and W_2. For both of them we have given a flat embedding as well. Let

$$A_{i,1}, A_{i,2}, \ldots, A_{i,l_i}, \text{ for } i = 1, 2,$$

denote the corresponding attached vertex sets. By Theorem 5.1.1, we can test in linear time if for $j = 1$ or $j = 2$ all subgraphs $G[A_i]$ with $i = 1, \ldots, l_i$ have tree-width at most $h(k)$. If so, the conditions for Theorem 5.2.28 are satisfied.

Otherwise, we may assume that $tw(G[A_{j,1}]) > h(k)$ holds for $j = 1, 2$. As $A_{1,1}$ and $A_{2,1}$ are disjoint, one of these two sets, let us say $A_{1,1}$, has cardinality at most $|V|/2$. Furthermore, $A_{1,1}$ is one of the components of $G - X - N(A_{1,1})$.

Now we start all over with $G[A_{1,1} \cup X \cup N(A_{1,1})]$ and repeat the previous procedure. When we come to the point of choosing two subwalls of height $f(k)$ there will exist two subwalls that do not contain any vertex of $X \cup N(A_{1,1})$, since we have

$$|X \cup N(A_{1,1})| \leq \binom{3k}{2} + 3 < 9k^2 - 2.$$

The next iteration step will only consider $G[A_{1,1}]$, hence reduce the size of the input graph by half.

From this it follows that the overall running time is of order $\mathcal{O}(n)$. To see this, denote the running time of the above procedure by $T(n)$. The running times of the individual iterations sum up to

$$T(n) + \mathcal{O}(n) + T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + \ldots,$$

which is of order $\mathcal{O}(n)$. □

5.2.1.3 The algorithm

The algorithm roughly runs as follows. First, a $2k$-certificate is computed and the number of edges is reduced to $\mathcal{O}(n)$ by repeatedly applying Theorem 5.2.20 and Theorem 5.2.24. If the k disjoint paths have not yet been found, a test for tree-width determines if it is possible to solve the problem directly by Theorem 5.2.26. If not, we can either find a suitable $3k$-clique minor or remove at least one irrelevant vertex. The latter produces a smaller input graph for the next iteration step. The algorithm terminates when the desired k paths have been found or when Theorem 5.2.26 returns that this is not possible.

Algorithm 5.2.30 (k disjoint paths problem)

INPUT: A graph $G = (V, E)$ with $|V| = n$ and terminals $s_1, \ldots, s_k, t_1, \ldots, t_k$

OUTPUT: **either** k disjoint paths $P_1, \ldots, P_k$ in G, each P_i connecting s_i and t_i
or the answer that this is not possible.

DESCRIPTION: The algorithm consists of four steps.

STEP 1. Use the algorithm from Theorem 5.2.19 to compute a $2k$-certificate G_{2k}. Then go to Step 2.

STEP 2. **While** $|\mathbf{E}| \geq \mathbf{2^{3k-3}|V|}$, apply Theorem 5.2.20 to find a $3k$-clique minor. Then, by Theorem 5.2.24,
either solve the problem and terminate
or remove some vertices and update G and G_{2k}.
Then go to Step 3.

STEP 3. Set $h(k)$ as in Theorem 5.2.29 and test for tree-width at most $h(k)$.
If $\mathbf{tw(G) \leq h(k)}$, use the tree-decomposition obtained by Theorem 5.1.1 to solve the problem by applying Theorem 5.2.26.
Else go to Step 4.

STEP 4. Apply Theorem 5.2.29.
Either a $3k$-clique minor is obtained, then utilise Theorem 5.2.24 to either terminate or remove vertices.

Or use (X, W) and Theorem 5.2.28 to find an irrelevant vertex and remove it.

Then go to Step 3.

RUNNING TIME: $\mathcal{O}(n^2)$

Proof (of running time).

STEP 1 takes time $\mathcal{O}(m)$, which is of order $\mathcal{O}(n^2)$.

STEP 2 is executed at most $\mathcal{O}(n)$ times. It takes $\mathcal{O}(n)$ time to find a $3k$-clique minor by Theorem 5.2.20 and Theorem 5.2.24 runs in at most $\mathcal{O}(n + \frac{m}{n})$ time. Hence, we have a total running time of $\mathcal{O}(n^2 + m)$.

STEP 3 tests for a tree-width of at most $h(k)$. This is possible in $\mathcal{O}(n)$ time by Theorem 5.1.1 which already returns a suitable tree-decomposition if it exists. As the number of edges is of order $\mathcal{O}(n)$, Theorem 5.2.26 takes $\mathcal{O}(f(k, h(k))n)$ time.

STEP 4 applies Theorem 5.2.29 in $\mathcal{O}(n)$ time. Solving the problem by Theorem 5.2.24 or finding an irrelevant vertex by Theorem 5.2.24 or Theorem 5.2.28 takes $\mathcal{O}(n)$ time. This is due to the fact that we work with a graph with a number of edges in $\mathcal{O}(n)$.

In total, the algorithm removes at most $\mathcal{O}(n)$ vertices from G. As this is an upper bound for the number of iterations, we get a total running time of $\mathcal{O}(n^2)$. □

5.2.1.4 Remarks

It should be mentioned that the running time of order $\mathcal{O}(n)$ depends, amongst others, on an algorithm by Kapadia et al. presented in [27]. [27] is a paper under submission. Without this result the running time of the algorithm by Kawarabayashi et al. would increase to $\mathcal{O}(n^2\alpha(n, n))$, where α is the inverse of the Ackermann function. This comes from an alternative test for flatness by Tholey which has an $\mathcal{O}(m + n\alpha(n, n))$ running time [54].

Second, the algorithm found by Robertson and Seymour is highly impractical. The constants hidden in the $\mathcal{O}$-notation are computable but huge. Moreover, in Theorem 5.2.29 some of those functions depending only on k are even combined to give a possible lower bound for $h(k)$. Since the algorithm of Kawarabayashi et al. is based on the results of Robertson and Seymour, it suffers from the same shortcomings. So, even if improvements have been made, it remains a challenging problem to further reduce the complexity of the general k disjoint paths problem.

5.3 Algorithmic implications for $\mathcal{NP}$-hard problems

The work of Robertson and Seymour had an enormous impact on the study of problems of unknown or $\mathcal{NP}$-hard complexity. On the one hand, their non-constructive proof of the existence of a finite list of excluded minors for every minor-closed class of graphs made it possible to deduce polynomial time decidability of problems for which no or only complex algorithms were previously known. It implies that polynomial time decidability of a problem P can be shown simply by proving that the class $\mathcal{C}_P$ of graphs satisfying P is minor-closed.

Amongst others, Fellows and Langston studied non-constructive tools for proving polynomial time decidability and how they could be made constructive [19, 18]. They also presented some problems which can be shown to be polynomial. Examples of such problems are linkless and knotless embeddability of graphs in 3-space, the gate matrix layout problem, the vertex cover problem and the longest path problem.

On the other hand, many $\mathcal{NP}$-hard problems can be shown to be polynomial time decidable when restricted to graphs of a certain regular structure, like graphs of bounded tree-width.

Bern, Lawler and Wong presented a systematic approach for solving a subgraph containment problem on a graph G satisfying certain composition properties [7]. There a subgraph corresponding to an optimum solution for a problem on G was sought. Problems to which this approach is applicable (on certain classes of graphs) include the minimum dominating set problem and the travelling salesman problem.

Furthermore, Courcelle and Arnborg et al. studied graph properties which can be formulated in monadic second-order logic. Problems expressible in such a way can be decided in linear time for graphs of bounded tree-width. See [3] and, for example, the series of papers starting with [15].

Arnborg and Proskurowski showed that many $\mathcal{NP}$-hard problems like finding a maximal independent set in a graph or colouring a graph can be solved in polynomial time when restricted to partial k-trees [4]. They used a dynamic programming approach computing partial solutions for subgraphs of a graph G induced by the partial k-tree structure of G. An analogous approach for graphs given together with a tree-decomposition was discussed by Bodlaender in [10]. This technique is presented in this chapter.

5.3.1 A generic approach for graphs of bounded tree-width

We want to examine a technique to solve a problem P on a graph G of bounded tree-width. If G is of relatively small width k, this technique helps to solve P efficiently by exploiting the connectivity properties of a tree-decomposition $(T, \mathcal{V})$ of width k of G. These properties allow us to take a dynamic programming approach where partial solutions are computed while traversing T. By using a tree-decomposition of a specific structure we simplify the design of our algorithms and the proofs of their correctness as well. This might also reduce the constant factors of running times of such algorithms.

5.3.1.1 Nice tree-decompositions

Definition 5.3.1 A *nice tree-decomposition* of a graph $G = (V, E)$ is a tree-decomposition $(T, \mathcal{V})$ of G such that

(1) T is a rooted tree and

(2) every $t \in V(T)$ is of one of the following four types:

Leaf node: t is a leaf in T and $|V_t| = 1$.

Introduce node: t has exactly one successor t' in T and $V_t = V_{t'} \cup \{v\}$ holds for some vertex $v \in V \setminus V_{t'}$.

Forget node: t has exactly one successor t' in T and $V_{t'} = V_t \cup \{v\}$ holds for some vertex $v \in V \setminus V_t$.

Join node: t has exactly two successors t_1 and t_2 in T and $V_t = V_{t_1} = V_{t_2}$ holds.
From now on, we will call the vertices of T *nodes*, consistent with the definitions above.

Given a tree-decomposition $(T, \mathcal{V})$ of width k of a graph G, it is possible to transform it into a nice tree-decomposition of G of same width k in linear time [10].

In the following, for a tree-decomposition $(T, \mathcal{V})$ and a vertex $t \in V(T)$, we denote by T_t the set of all vertices

$$\{t' \in V(T) \mid t' \geq_T t\}$$

of T succeeding t in the tree order induced on T by a fixed root $r(T)$. T_t is the subtree of T rooted at t. We additionally set

$$Y_t = \bigcup_{t' \in T_t} V_{t'} \quad \text{and} \quad G_t = G[Y_t].$$

5.3.1.2 Main steps of the generic algorithm

Suppose we have given a graph $G = (V, E)$ of tree-width $tw(G)$ bounded from above by an integer k. Our approach to decide a given problem on G generally executes the following steps:

(1) A tree-decomposition of G of width at most k is computed. We know from Theorem 5.1.1 that this is possible in linear time.
(2) This tree-decomposition is used to construct a nice tree-decomposition $(T, \mathcal{V})$ whose width is also bounded from above by k.
(3) T is traversed in postorder (with respect to its tree order), that is, from the leaves to an arbitrarily fixed root r(T). For every node $t \in V(T)$ a partial solution to our problem is computed for G_t. This step combines information about the partial solutions computed for the successors of t with local information about the vertices in V_t. Depending on the type of node t, a predefined computing scheme is employed.
(4) The (partial) solution computed for the root $r(T)$ then yields an answer to the global problem given for G.

This procedure applies to a decision problem on G. The corresponding search problem can be solved with an additional step. In this step a solution is constructed by using information obtained in the previous steps.

5.3.1.3 The necessary components in more detail

For every problem P given for a graph G, we have to define certain notions and techniques specifically adapted to P.

This includes the exact definition of a *solution* for G and a *partial solution* for a subgraph of G. In our case, this is always a subgraph of the form G_t for a node $t \in V(T)$. A solution for G should always provide a partial solution for each subgraph G_t of G. It is also necessary to define a valid *extension* of a partial solution to a supergraph.

For every node $t \in V(T)$ we define the *characteristic* of a partial solution for G_t. It concentrates the information about the partial solution needed for the rest of the procedure. The *full set of characteristics for* G_t, or *full set for* t, consists of the different characteristics of all partial solutions for G_t.

It it also important to assure that a full set for a node t can be computed efficiently if the full sets for all successors of t in T are given. And, additionally, that a full set for the root $r(T)$ is sufficient to efficiently decide P for G.

Therefore, it is necessary to define for every type of node how the full sets for nodes of this type are computed. This yields a construction scheme combining the full sets for the existing successors with local information about the node considered. It is important that this scheme can be implemented efficiently. Often, good upper bounds can be obtained due to the bound given for the tree-width of G.

5.3.1.4 An example: l-colouring of vertices

To illustrate the information given above we examine the decision problem for a valid l-colouring of the vertices of a given graph $G = (V, E)$ of bounded tree-width $tw(G) \leq k$. That is, deciding if it is possible to colour each vertex $v \in V$ with one of l colours such that no edge in E has two endvertices coloured with the same colour. We define an l-colouring of G as a function

$$f : V \to \{1, \ldots, l\}$$

mapping every vertex to one of l numbered colours.

Let $(T, \mathcal{V})$ be a nice tree-decomposition of G of width at most k. A *solution* for G is an l-colouring f of G such that

$$\forall vw \in E : f(v) \neq f(w).$$

We call this a *valid* l-colouring of G. A *partial solution* f for a subgraph G_t of G is a solution for G_t. It can be *extended* to a supergraph G' of G_t if there exists a valid l-colouring g of G' with $g|_{V_t} = f$.

For a node $t \in V(T)$ the *characteristic* of a partial solution f for G_t is the restriction $f|_{V_t}$ of f to the set V_t. Therefore, if two partial solutions have the same characteristic one of them can be extended to a solution for G if and only if the other can be extended as well.

The *full set* for t is the set of all possible valid l-colourings of V_t. As we have $|V_t| \leq k + 1$, the full set for t contains at most l^{k+1} possible l-colourings, which is a constant for k and l fixed. For each type of node (defined as above), we now describe how to compute the full set.

Leaf node t: The set of partial solutions for $V_t = \{v\}$ simply consists of the l possible colourings of v, as does the full set for t.

Introduce node t: The partial solutions for the single successor t' of t with $V_t = V_{t'} \cup \{v\}$ have already been computed when t is visited. Since v is not contained in $V_{t'}$ its only neighbours in G_t are vertices from V_t. We simply need to determine for every characteristic in the full set for t' with which possible colourings of v it is compatible. Therefore the full set for t consists of all such combinations yielding a valid l-colouring of V_t.

Forget node t: All information needed to compute the full set for t is already implied in the full set for the only successor t' in T. Since $V_{t'} = V_t \cup \{v\}$ holds for some vertex $v \in V \setminus V_t$ we simply have to restrict the given characteristics to V_t. Every valid l-colouring of $G_{t'}$ is a valid l-colouring of G_t as well.

Join node t: We have $V_t = V_{t_1} = V_{t_2}$ for the two successors t_1 and t_2 of t. The full set of characteristics for t therefore corresponds to the intersection of the full characteristics given for t_1 and t_2.

Hence, if the full set for the root $r(T)$ is non-empty we know that there exists at least one valid l-colouring of the vertices of G.

Moreover, we see that the complexity of our algorithm might be exponential in k but is linear in the order of T and therefore in $|V|$. As many real-world applications handle graphs of huge size but small tree-width, an approach similar to the one presented above can have significant advantages in practice.

5.4 Graph searching

In this section we present a possibility to use already introduced concepts like tree-width and brambles for a game on graphs. We will focus on a *pursuit-evasion game* discussed by Seymour and Thomas in [53].

In this game, a fixed number k of cops search for a robber that is hiding in the vertices of a graph G. Let C denote the set of vertices occupied by cops. The cops always know in which component R of $G - C$ the robber lingers and direct their steps towards him. They travel by helicopter, so at all times a cop is either occupying a vertex of G or in a helicopter. A move of the cops consists of some cops getting on a helicopter or of some cops returning to the graph. More formally, after their move we have a new set of vertices C' guarded by cops such that $C' \subseteq C$ or $C \subseteq C'$ holds. The robber can try to evade the cops by running at any time to another vertex that he can reach from his current position along a path in G on which no vertex is taken by a cop. Therefore it is only of interest in which component R' of $G - C'$ he resides. R' has to fulfil either $R \subseteq R'$ or $R' \subseteq R$.

Definition 5.4.1 We say that the graph G can be *searched by* $< k$ cops if $k - 1$ cops suffice to corner any robber at some vertex and then capture him. If they never have to search any vertex twice, then $< k$ cops can *monotonously search* the graph G. On the other hand, if there exists an escape strategy for the robber such that the cops cannot guarantee to catch him (with a monotone search strategy), we say that $< k$ cops *cannot (monotonously) search* G.

Definition 5.4.2 A graph G has *search number* n if n is minimal with the property that n cops can search G.

We look at a small example (taken from [8]). The graph in fig. 5.2 has search number 3. Two cops are not sufficient to corner the robber and also capture it. He can always hide in one of the vertices b, c and d. The tree-decomposition gives a strategy to monotonously search the graph with 3 cops.

In [53], Seymour and Thomas showed that if $< k$ cops can search a graph G, they can also monotonously search it. And that, if a robber is able to elude capture, we can easily describe his escape strategy using brambles. Even if we give the cops more flexibility and allow them to *jump-search* the graph G.

Definition 5.4.3 *Jump-searching* differs from the searching method already described only in that in one step the $k - 1$ cops can arbitrarily choose their next position C' from

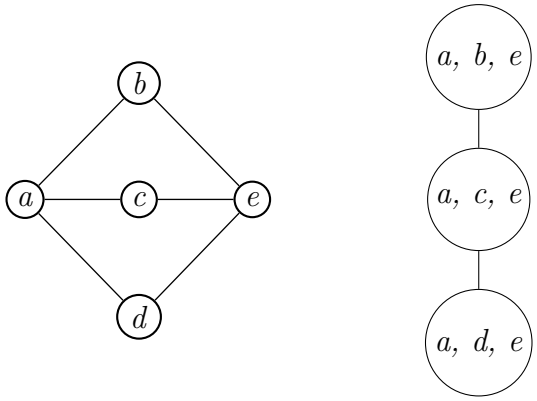

Figure 5.2: graph with search number 3 and a tree-decomposition of width 2

$[V(G)]^{<k}$. Then, as before, the robber may flee to a component R' of $G - C'$ that touches his last refuge R.

We will try to describe the ideal escape plan of a robber in a graph G with a function that tells him in which component to hide.

Lemma 5.4.4 *A graph $G = (V, E)$ cannot be jump-searched by $< k$ cops if and only if there is a function σ mapping each $C \in [V]^{<k}$ to a non-empty union $\sigma(C)$ of components of $G - C$, such that every component R in $\sigma(C)$ touches $\sigma(C')$ for all $C, C' \in [V]^{<k}$.*

Proof. If there is such a function σ and the robber chooses for every placement C of the cops a component from $\sigma(C)$ to hide, he will always find a new hiding place after the following step of the cops by choosing a component touching the component he is currently in.

Now, if $< k$ cops cannot jump-search the graph, there will be configurations (C, R) such that $k - 1$ cops cannot guarantee to win. Let $\sigma(C)$ consist of all components R of $G - C$ with this property. □

Definition 5.4.5 Let $G = (V, E)$ be a graph. A *haven of order k in G* is a function τ that assigns to every set C in $[V]^{<k}$ a component $\tau(C)$ of $G - C$ such that $\tau(C)$ touches $\tau(C')$ for all $C' \in [V]^{<k}$.

Interestingly, the brambles in G pose a possibility of finding a haven in G, hence we can use their properties to draw conclusions about graph searching.

Lemma 5.4.6 *A graph $G = (V, E)$ contains a haven of order $\geq k$ if and only if G contains a bramble of order $\geq k$.*

Proof. Let τ be a haven of order k in G. Then the set $\mathcal{B}$ of all sets $V(\tau(C))$ with C in $[V]^{<k}$ forms a bramble of order k. This is due to the fact that for every potential cover S of $\mathcal{B}$ with cardinality $< k$ there exists at least one member of $\mathcal{B}$, namely $\tau(S)$, not covered by S.

Every bramble $\mathcal{B}$ naturally causes the existence of a haven τ in G. For every set $C \in [V]^{<k}$ there exists a $B \in \mathcal{B}$ not covered by C. Set $\tau(C)$ to be the component of $G - C$ containing B. As the sets of $\mathcal{B}$ are mutually touching, we obtain a haven of order $\geq k$. □

Combining the past two lemmata with our knowledge about brambles results in the following main statement of [53].

Theorem 5.4.7 *For a graph G and an integer $k \geq 1$ the following are equivalent:*
(1) G contains a bramble of order $\geq k$.
(2) G contains a haven of order $\geq k$.
(3) $< k$ cops cannot jump-search G.
(4) $< k$ cops cannot search G.
(5) $< k$ cops cannot monotonously search G.
(6) G has tree-width $\geq k - 1$.

Proof. According to Lemma 5.4.6 the first two statements are equivalent and Lemma 5.4.4 tells us that (3) follows from (2). It is obvious that (3) implies (4) which in turn implies (5).

Now assume that G has tree-width $tw(G) < k - 1$ and let $(T, \mathcal{V})$ be a corresponding tree-decomposition. Then, with at most $k-2$ cops we can start by occupying all vertices contained in one part V_{t_1} from $\mathcal{V}$. Let V_{t_2} be its neighbour in the component where the robber hides. V_{t_2} is the next choice of placement for the cops. As $V_{t_1} \cap V_{t_2}$ separates G, the cops can proceed in this manner to corner the robber in smaller and smaller segments of G and finally capture him without returning to a vertex a second time. Therefore we have (5) $\Rightarrow$ (6).

The missing implication (6) $\Rightarrow$ (1) is given by Theorem 4.2.5, hence that a tree-width of $tw(G) \geq k - 1$ implies a bramble number $\beta(G)$ of at least k. □

Corollary 5.4.8 *Let $sn(G)$ denote the search number of a graph G. Then we have*

$$sn(G) = \beta(G) = tw(G) + 1.$$

Chapter 6

Related Work and Further Research

The intention of this chapter is to give some additional information about the topics touched in this thesis and rounding it off by referring to related research and open questions.

6.1 The graph minor theorem

In this thesis, we focused on the case of excluding a planar graph. This is the first part of the proof of the graph minor theorem. Now, we roughly sketch the complete proof.

Suppose we have given a countable sequence of graphs

$$G_0, G_1, G_2, \ldots$$

and suppose that no graph G_j with index $j \geq 1$ contains G_0 as a $\lesssim$-minor, otherwise we have already found a good pair. Thus, all these graphs are elements of $Forb(G_0)$. We have settled the case that G_0 is planar in Theorem 4.1.1. Robertson and Seymour showed for every non-planar graph H that every graph G not containing H as a $\lesssim$-minor can be constructed by taking clique-sums of graphs which are "nearly embeddable" in surfaces in which H is not embeddable [46]. A technique similar to the one introduced by Nash-Williams for the proof of Kruskal's Theorem 2.2.9 on trees is then used to find a good pair.

While for the first part of the proof several simplifications have been found, the same does not hold for the general case. There do exist simpler proofs for the generalisation of Kuratowski's Theorem 2.2.8 that we mentioned in section 4.5. But still it would be of great interest to find simplifications for all parts of the proof. Moreover, a constructive approach would probably have considerable advantages in practice.

A good introduction to graph minor theory and the graph minor theorem in particular can be found in Diestel's textbook on graph theory [16]. Kawarabayashi and Mohar wrote a survey with a focus on the excluded minor theorem for a general graph and some of its variations [30]. Other surveys on the graph minor theorem and its implications are [8, 35].

One open question concerning countably infinite graphs worth mentioning was formulated by Seymour.

Conjecture 6.1.1 (Seymour's self-minor conjecture) *Every countably infinite graph is a proper minor of itself.*

6.2 On computing tree-width

We already know that the general problem of computing tree-width is $\mathcal{NP}$-hard [2]. For fixed k, Theorem 5.1.1 states the existence of a linear time algorithm to compute the tree-width of a graph of tree-width at most k.

Designing fast algorithms computing tree-width exactly is a challenging problem. It is even an open question if there exist algorithms approximating tree-width within a constant factor [21]. Concerning exact algorithms for tree-width, the best bound currently known for the running time is $\mathcal{O}(1.7347^n)$ which is based on a paper of Fomin and Villanger [20]. However, the implementation of this algorithm may prove to be difficult due to necessary computations of potential maximal cliques [13]. Additionally, this algorithm requires an exponential amount of space. Algorithms using only polynomial space exist as well. Here, we refer to [13] and [21] for exact algorithms using polynomial space and having a running time of order $\mathcal{O}(2.9512^n)$ and $\mathcal{O}(2.6151^n)$, respectively. Still, they are of little practical use.

With regard to this thesis, we want to remark that the different characterisations given in section 3.3 prove to be useful for the computation of tree-width. Many algorithms exploit the close relation between tree-width and elimination orderings. Other approaches utilise minimal separators.

As we have seen in section 4.2, brambles can be employed to find lower bounds for the tree-width of a graph. Computing the order of a given bramble is, however, $\mathcal{NP}$-hard. Nevertheless, Bodlaender states that this approach can be useful for planar graphs or graphs that are close to being planar [12].

The possibility of approximating the tree-width of a planar graph within a factor of 3/2 in polynomial time was assured by Seymour and Thomas. They showed that the computation of branch-width is polynomial for planar graphs but $\mathcal{NP}$-hard for general graphs [52].

6.3 On graph searching

In section 5.4 we presented only the pursuit-evasion game studied by Seymour and Thomas in [53]. However, there exist many other similar games and problems.

If, for instance, the game is only changed inasmuch that the robber is invisible the search number would significantly rise: We know that two cops are able to monotonously search a tree for a visible robber and finally arrest him. If they have no information about his hiding place they can not guarantee to ever catch him.

There also exist differences concerning the movement of the robbers and how they can make sure that some portion of the graph is "robber-free". In the case of an invisible robber, this game is also interpreted as clearing an entirely contaminated graph from a plague, or immunizing a network against a computer virus. *Edge searching* means that the cops expand the cleared portion of a graph by sliding along an edge. When

node searching, they clear an edge by occupying both its endvertices. There also exist *mixed searching* versions of this game. It can be shown that the search number for node searching equals the pathwidth of the graph considered plus one.

Seymour and Thomas give a quick overview on related research in the field of graph searching [53]. Other surveys can be found in [12] and [8].

6.4 Further research

As we have already mentioned, one major topic of interest is simplifying the proof of the graph minor theorem and making it more accessible.

Furthermore, it is necessary to reduce the actual running time of algorithms related to the topics touched in this thesis. Huge constants often render polynomial algorithms unemployable in practice. Examples are the algorithms presented for computing tree-width or solving the k disjoint paths problem. For some thoughts on complexity concerning questions related to the work of Robertson and Seymour see for example [26].

We conclude with one last example concerning the implications of the graph minor theorem for minor-closed classes of graphs. Showing that a graph property is inherited by minors gives a deep insight into the complexity of membership testing for that class. Moreover, if some planar graph is excluded as a $\lesssim$-minor then we have even more knowledge about the graphs contained in this class. We can further draw conclusions about the complexity of some algorithms on graphs in this class. Therefore, the search for minor-closed classes of graphs and also for some of their possibly yet unknown properties will have its benefits.

Bibliography

[1] Isolde Adler, Frederic Dorn, Fedor V. Fomin, Ignasi Sau, and Dimitrios M. Thilikos. Fast minor testing in planar graphs. *Algorithmica*, 64(1):69–84, 2012.

[2] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[3] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.

[4] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Appl. Math.*, 23(1):11–24, 1989.

[5] Patrick Bellenbaum and Reinhard Diestel. Two short proofs concerning tree-decompositions. *Combin. Probab. Comput.*, 11(6):541–547, 2002.

[6] Horst Bergmann. Ein Planaritätskriterium für endliche Graphen. *Elem. Math.*, 37(2):49–51, 1982.

[7] M. W. Bern, E.L. Lawler, and A. L. Wong. Why certain subgraph computations requite only linear time. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 117–125, 1985.

[8] Daniel Bienstock and Michael A. Langston. Algorithmic implications of the graph minor theorem. In *Network models*, volume 7 of *Handbooks Oper. Res. Management Sci.*, pages 481–502. North-Holland, Amsterdam, 1995.

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

[10] Hans L. Bodlaender. Treewidth: algorithmic techniques and results. In *Mathematical foundations of computer science 1997 (Bratislava)*, volume 1295 of *Lecture Notes in Comput. Sci.*, pages 19–36. Springer, Berlin, 1997.

[11] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1-2):1–45, 1998.

[12] Hans L. Bodlaender. Treewidth: structure and algorithms. In *Structural information and communication complexity*, volume 4474 of *Lecture Notes in Comput. Sci.*, pages 11–25. Springer, Berlin, 2007.

[13] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. On exact algorithms for treewidth. *ACM Trans. Algorithms*, 9(1):Art. 12, 23, 2012.

[14] Béla Bollobás. *Modern graph theory*, volume 184 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.

[15] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990.

[16] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fourth edition, 2010.

[17] Reinhard Diestel, Tommy R. Jensen, Konstantin Yu. Gorbunov, and Carsten Thomassen. Highly connected sets and the excluded grid theorem. *J. Combin. Theory Ser. B*, 75(1):61–73, 1999.

[18] M. R. Fellows and M. A. Langston. On search decision and the efficiency of polynomial-time algorithms. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, STOC '89, pages 501–512, New York, NY, USA, 1989. ACM.

[19] Michael R. Fellows and Michael A. Langston. Nonconstructive tools for proving polynomial-time decidability. *J. Assoc. Comput. Mach.*, 35(3):727–739, 1988.

[20] Fedor V. Fomin and Yngve Villanger. Finding induced subgraphs via minimal triangulations. In *STACS 2010: 27th International Symposium on Theoretical Aspects of Computer Science*, volume 5 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 383–394. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2010.

[21] Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

[22] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combinatorial Theory Ser. B*, 16:47–56, 1974.

[23] James F. Geelen, A. M. H. Gerards, and Geoff Whittle. Branch-width and well-quasi-ordering in matroids and graphs. *J. Combin. Theory Ser. B*, 84(2):270–290, 2002.

[24] Rudolf Halin. S-functions for graphs. *J. Geometry*, 8(1-2):171–186, 1976.

[25] Illya V. Hicks, Sang-il Oum, James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith. *Branch-Width and Tangles*. John Wiley & Sons, Inc., 2010.

[26] David S. Johnson. The NP-completeness column: an ongoing guide. *J. Algorithms*, 8(2):285–303, 1987.

[27] R. Kapadia, Li Z., and B. Reed. Two disjoint rooted paths in linear time. *preprint*.

[28] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

[29] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *J. Combin. Theory Ser. B*, 102(2):424–435, 2012.

[30] Ken-ichi Kawarabayashi and Bojan Mohar. Some recent progress and applications in graph minor theory. *Graphs Combin.*, 23(1):1–46, 2007.

[31] Ken-ichi Kawarabayashi and Paul Wollan. A shorter proof of the graph minor algorithm: the unique linkage theorem. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 687–694, New York, NY, USA, 2010. ACM.

[32] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Pearson Addison-Wesley, Boston, 2010.

[33] J. B. Kruskal. Well-quasi-ordering, the Tree Theorem, and Vazsonyi's conjecture. *Trans. Amer. Math. Soc.*, 95:210–225, 1960.

[34] Casimir Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930.

[35] László Lovász. Graph minor theory. *Bull. Amer. Math. Soc. (N.S.)*, 43(1):75–86 (electronic), 2006.

[36] Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5-6):583–596, 1992.

[37] C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Proc. Cambridge Philos. Soc.*, 59:833–835, 1963.

[38] Ljubomir Perković and Bruce Reed. An improved algorithm for finding tree decompositions of small width. *Internat. J. Found. Comput. Sci.*, 11(3):365–371, 2000. Selected papers from the Workshop on Theoretical Aspects of Computer Science (WG 99), Part 1 (Ascona).

[39] B. A. Reed, N. Robertson, A. Schrijver, and P. D. Seymour. Finding disjoint trees in planar graphs in linear time. In *Graph structure theory (Seattle, WA, 1991)*, volume 147 of *Contemp. Math.*, pages 295–301. Amer. Math. Soc., Providence, RI, 1993.

[40] Bruce Reed and David R. Wood. A linear-time algorithm to find a separator in a graph excluding a minor. *ACM Trans. Algorithms*, 5(4):39:1–39:16, November 2009.

[41] Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[42] Neil Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B*, 41(1):92–114, 1986.

[43] Neil Robertson and P. D. Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Combin. Theory Ser. B*, 48(2):227–254, 1990.

[44] Neil Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.

[45] Neil Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995.

[46] Neil Robertson and P. D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *J. Combin. Theory Ser. B*, 89(1):43–76, 2003.

[47] Neil Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Combin. Theory Ser. B*, 92(2):325–357, 2004.

[48] Neil Robertson, Paul Seymour, and Robin Thomas. Quickly excluding a planar graph. *J. Combin. Theory Ser. B*, 62(2):323–348, 1994.

[49] Donald J. Rose. Triangulated graphs and the elimination process. *J. Math. Anal. Appl.*, 32:597–609, 1970.

[50] Donald J. Rose. On simple characterizations of k-trees. *Discrete Math.*, 7:317–322, 1974.

[51] P. D. Seymour. Disjoint paths in graphs. *Discrete Math.*, 29(3):293–309, 1980.

[52] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[53] P. D. Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *J. Combin. Theory Ser. B*, 58(1):22–33, 1993.

[54] Torsten Tholey. Improved algorithms for the 2-vertex disjoint paths problem. In *SOFSEM 2009: theory and practice of computer science*, volume 5404 of *Lecture Notes in Comput. Sci.*, pages 546–557. Springer, Berlin, 2009.

[55] J. van Leeuwen. Graph algorithms. In *Handbook of theoretical computer science, Vol. A*, pages 525–631. Elsevier, Amsterdam, 1990.

[56] K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, 114(1):570–590, 1937.