Graph minors, decompositions and algorithms

(Lecture notes)

Jiří Fiala

Department of Applied Mathematics, Charles University, Prague

fiala@kam.mff.cuni.cz

April 15, 2024

For updates see http://kam.mff.cuni.cz/~fiala/tw.pdf.
Course instructors, write me for a version with all solutions.

Contents

1 Introduction

In this lecture notes we would like to explore some relationship between the notions of *graph minors* and *treewidth*. After the course, the student should be familiar enough with these notions to be be capable to apply them in the design of graph algorithms.

Some hard theorems in this course will be provided without proofs (the complete proof of Robertson-Seymour theorem has been given in more than twenty papers). In simpler cases proofs will be given, to distinguish hard and easy situations in this theory.

Students are welcome to solve exercises at the end of each lecture. Many exercises are taken from [39]. Some straightforward exercises are inserted into the text for better understanding of the subject.

A substantial part of these lecture notes is taken from the monograph [12]. An interested reader can find there more details on the subject of graph minors and tree decompositions.

1.1 Basic definitions and properties of graph minors

If not specified otherwise, all graphs we consider are finite, undirected, simple and loopless. It means that a graph G is given as a pair (V, E), where V is a finite set of vertices and E is a set of unordered pairs which we call edges. Formally: $G = (V, E),\ E \subseteq \binom{V}{2}$. We use the notation (u, v) to denote the undirected edges, while directed edges will be written as $[u, v]$.

If loops and multiple edges are present, the structure is called *multigraph*.

A subgraph $H = (V_H, E_H)$ of a graph $G = (V_G, E_G)$, is a graph[1] that satisfies $V_H \subseteq V_G$ and $E_H \subseteq \binom{V_H}{2} \cap E_G$. If $E_H = \binom{V_H}{2} \cap E_G$, we say that H is a subgraph of G induced by the set V_H, or simply an induced subgraph. In such a case we write $H = G|_{V(H)}$.

[1]More precisely we would say that H is isomorphic to such graph. For simplicity we will avoid this formalism. Similarly, we will not deal in details with the empty graph $(\emptyset, \emptyset)$ as the universal subgraph.

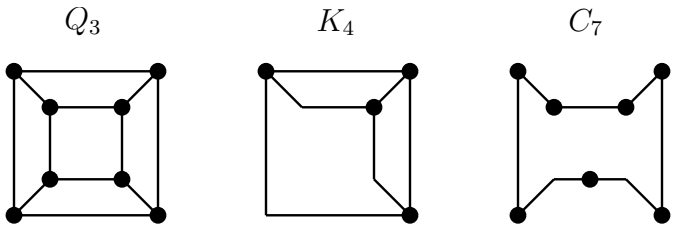


Figure 1: Examples of graph minors

Let G be a graph G, $w \in V_G$ be a vertex of G and $e = (u, v) \in E_G$ be an edge. We define the following operations:

- The *deletion* of the vertex w results into the graph $G - w$, which is the subgraph of G induced by the set $V_G \setminus \{w\}$.
- The *deletion* of the edge e gives the graph $G - e = (V_G, E_G \setminus \{e\})$.
- The *subdivision* of e yields the graph $G \cdot e$, where $V(G \cdot e) = V_G \cup \{v' : v' \notin V_G\}$ and $E(G \cdot e) = E_G \setminus \{e\} \cup \{(u, v'), (v, v')\}$.
- The *contraction* of e provides the graph $G \circ e$ with the vertex set $V(G \circ e) = V_G \setminus \{v\}$ and edges $E(G \circ e) = E_G \setminus \{e : v \in e\} \cup \{(u, v') : (v, v') \in E_G, v' \neq u\}$.
- We say that a contraction $G \circ e$ is *topological*, if at least one vertex of e has degree two in G. Such a contraction can be viewed as an inverse operation to the subdivision operation.

Now we have enough knowledge to give the first nontrivial definition:

Definition. A graph H is a *minor* of a graph G, if H can be obtained from a subgraph of G by a sequence of edge contractions. If H was derived from an induced subgraph of G, we call H an *induced minor*. We say that H is a *topological minor* of G, if all contractions used in the transformation of G into H were topological contractions.

In other words H is a topological minor of G, if G contains a *subdivision* of H as a subgraph, i.e., a graph which can obtained from H by a series of subdivision operations.

An example is depicted in Fig. 1. Here K_4 is an induced (and also a topological) minor of Q_3. Graph C_7 is a topological minor of Q_3, but not induced.

Definition. A class of graphs $\mathcal{G}$ is called *(induced) minor closed*, if it contains with every $G \in \mathcal{G}$ all its (induced) minors.

We provide an alternative definition of the notion of minor:

Lemma 1.1. *A graph H is an (induced) minor of G, if and only if there exists a mapping f from the vertex set of H to the set of subsets of vertices of G, such that:*

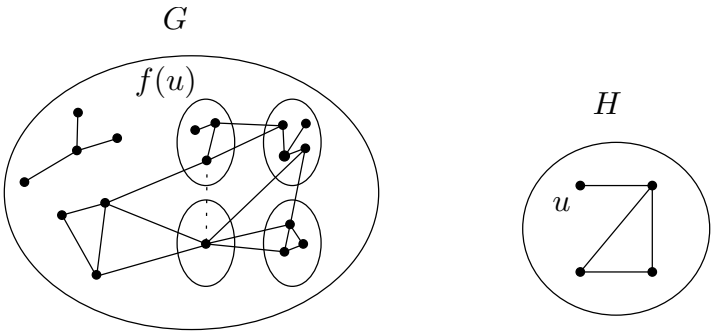


Figure 2: An illustration for Lemma 1.1. The dotted line should not be present, when H is an induced minor of G.

- *For every $u \in V_H$ the graph $G|_{f(u)}$ is nonempty and connected.*
- *For every $u, v \in V_H$ sets $f(u)$ and $f(v)$ are disjoint.*
- *If (and only if) there is an edge (u, v) in H, then there exists an edge (u', v') in G, such that $u' \in f(u), v' \in f(v)$.*

Proof. To get the mapping f: Assume that a vertex w of H is the result of contracting edges $e_1, \ldots, e_k$ of G. Then these edges form a connected subgraph of G. Moreover, these subgraphs are vertex-disjoint for different vertices w. Define f, s.t. $f(w)$ is the set of vertices incident with edges $e_1, \ldots, e_k$. We prove the third property by contradiction: If none (u', v') exists, then it is impossible to obtain the edge (u, v) by a series of contractions. For induced minors: if some (u', v') exists, then it is impossible to get rid of the edge (u, v), which must be present due to contractions.

In the opposite direction: having the mapping f, we first take the subgraph in G induced by $\bigcup_{u \in V_H} f(u)$. Now every $f(u)$ can be contracted into a single vertex, and analogously as $f(u)$ and $f(v)$ are disjoint, the contraction of $f(u)$ yields a different vertex than $f(v)$. So after these two steps we get an induced minor of G, which can be further reduced by edge removal to a minor isomorphic to H. □

We further link the notion of minor with the subdivision operation:

Lemma 1.2 ([31])**.** *If H is of maximum degree at most three, then H is a minor of G if and only if H is a topological minor of G.*

Proof. Only $\Rightarrow$ implication is sufficient to prove. Let G' be the subgraph of G whose contraction is isomorphic to H. Find a mapping f from Lemma 1.1. For every edge $e = (u, v) \in E_H$ identify an edge $f'(e) = (u', v') \in E_{G'}$, such that $u' \in f(u), v' \in f(v)$, and mark both vertices u' and v'. As H is of maximum degree three every $f(u)$ contains at most three marked vertices. In each $G'|_{f(u)}$ we find some inclusion-wise minimal connected subgraph containing all marked vertices. This graph is a tree of maximum degree at most three and selection of such trees for every $u \in V_H$, together with edges $\{f'(e) : e \in E_H\}$ creates the desired subdivision of H. □

Exercise 1: Show that the degree condition in Lemma 1.2 is necessary.

Exercise 2: Show that Lemma 1.2 does not hold for induced minors.

Let $V = \{u_1, \ldots, u_n\}$ be the vertex set. We define:

- the empty graph $E_n = (V, \emptyset)$, $n \geq 0$
- the complete graph (clique) $K_n = (V, \binom{V}{2})$, $n \geq 1$,
- the path $P_n = (V, \{(u_i, u_{i+1}), i = 1, \ldots, n-1\})$, $n \geq 2$,
- the cycle $C_n = (V, E(P_n) \cup \{(u_1, u_n)\})$, $n \geq 3$,

Exercise 3: Decide whether the class of all empty graphs, cliques, paths or cycles is (induced) minor-closed.

Exercise 4: Find the minimum (inclusion-wise) minor closed class of graphs that contains all paths, and all cycles, resp.

Two vertices $u, v \in V_G$ are connected in G if either $u = v$ or G has a path as a subgraph that contains both u and v. A graph G is called connected if every two vertices u, v are connected. Every inclusion-maximal subset of vertices that are mutually connected is called a component.

A forest is a graph which has no cycle. A connected forest is called a tree.

Exercise 5: Are connected graphs, or trees or forests (induced)-minor closed? Which subclass of connected graph is (induced)-minor closed?

Exercise 6: Which degree-bounded graphs are minor closed?

Almost-k-trees are subgraphs of connected graphs on n vertices with at most $n + k - 1$ edges.

Exercise 7: Show that forest are almost-0-trees.

Exercise 8: Show that unicyclic graphs (graph with at most one cycle) are almost-1-trees.

Exercise 9: Are unicyclic graphs and almost-k-trees (induced) minor-closed?

The intersection graph of a set system is defined as follows. Let X be a (finite) family of sets, then the *intersection graph* I_X of X has the vertex set $V(I_X) = X$, and edges connect the sets with nonempty intersection, i.e., $E(I_X) = \{(x, y) : x, y \in X, x \cap y \neq \emptyset\}$.

A string graph is the intersection graph of a set of curves in the plane. Interval graphs are intersection graphs of intervals of real line.

Exercise 10: Are string/interval graphs (induced) minor closed?

Line graphs are intersection graphs of edges of a graph.

Exercise 11: Are line graphs (induced) minor closed?

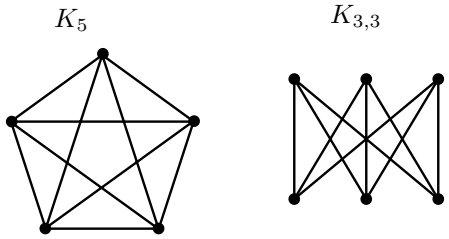


Figure 3: Forbidden minors for planar graphs.

1.2 Minors and graph drawing

Planar graphs are graphs that can be drawn in the Euclidean plane such that vertices are represented by points and edges are drawn as curves connecting adjacent vertices which intersect only at their endpoints.

Observe that the operations of taking subgraphs and contracting edges preserve the graph embedding in the plane. As this observation is valid for an arbitrary surface, it follows that for any surface S (including the Euclidean plane) the class of graphs that can be embedded on S is minor closed.

The class of planar graphs is well characterized by the celebrated Kuratowski theorem [18]. For a formal proof see e.g. [25, 12].

Theorem 1.3. *A graph is planar, if and only if it contains no subdivision of K_5 nor of $K_{3,3}$ as a subgraph.*

Idea of the proof [25]. We first restrict ourselves to 3-connected graphs as otherwise drawings of 2-connected blocks can be combined together (in order to have each 2-cut in the same face it is necessary to prove that each such cut can be assumed to be an edge.)

It is possible to assume that there exists an edge $e = (u, v)$, such that $G \circ e$ is still 3-connected (if G has at least 6 vertices). Assume that $G \circ e$ has a planar drawing where the boundary of each face is a convex polygon. We further assume that u and v are not incident with the outer face. Restrict the drawing of $G \circ e$ to $G - e$. Let further C be the cycle bounding the face containing the point that was representing e. The drawing of $G - e$ can be extended to a drawing of G with convex faces if the edges of C can be split in two paths, one containing the neigbors of u and the other neigbors of v (the endpoints may be neighbors of both).

If this is not possible, then either u and v have three common neighbors on C, which together with C provide a subdivision of K_5. Alternatively, there are distinct vertices u', v', u'', v'' that appear on this order on C, where u', u'' are neighbors of u, and analogously v', v'' are neighbors of v. Now u, v and C yield a subdivision of $K_{3,3}$. □

By the theorem, planar graphs are characterized by *forbidden subdivisions* (i.e., by forbidden topological minors). We show that planarity could be expressed in terms of forbidden minors:

Corollary 1.4. *A graph G is planar, if and only if none of K_5 and $K_{3,3}$ is its minor.*

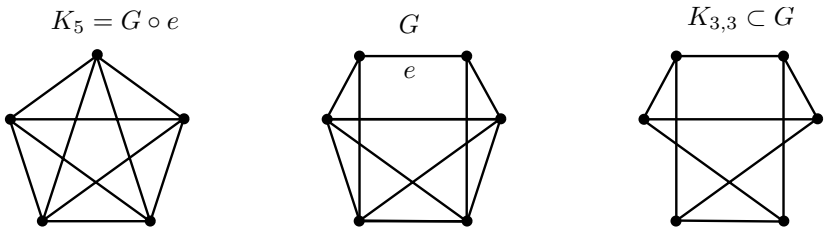


Figure 4: Obtaining $K_{3,3}$ minor from G.

Proof. If G is planar then it has no non-planar minor like K_5 and $K_{3,3}$. If G is not planar then it has a subdivision of K_5 or $K_{3,3}$ which yield a minor. □

Indeed the existence of a topological minor could be seen directly:

Observation 1.5. *If G has a K_5 or $K_{3,3}$ minor, then it has a K_5 or $K_{3,3}$ topological minor.*

Proof. As $\Delta(K_{3,3}) = 3$ we have due to Lemma 1.2 that if $K_{3,3}$ is a minor of G then it is also its topological minor.

Hence it suffices to show: *"If G has a K_5 minor, then it contains either K_5 or $K_{3,3}$ as a topological minor."*

Assume that K_5 was obtained by a series of contractions from a subgraph of G. If all contractions were topological, then K_5 is a topological minor of G.

Otherwise we have contracted some edges between vertices of degree at least three. Without loss of generality we may assume that it is the last contraction. The graph before the last contraction contains a $K_{3,3}$ minor (see Fig. 4). By Lemma 1.2 we may conclude that $K_{3,3}$ is a topological minor of G. □

To see that the existence of such characterization is a general principle, we leave for the next lecture.

We further explore a relation to dual drawings.

Definition. Let $G = (V, E)$ be a multigraph drawn on a surface S. Let F be a set of points chosen such that each face of the drawing of G is represented by a single point in F. Then the *dual* of G is the multigraph $G^* = (F, E^*)$ whose edges are given by the equivalence:

$$\begin{gathered} e = (u, v) \in E \text{ separates faces represented by points } g \text{ and } h \\ \Longleftrightarrow \\ e^* = (g, h) \in E^* \text{ separates faces represented by points } u \text{ and } v \end{gathered}$$

The drawing of G^* is given by drawing e^* such that it intersects only e and the adjacent faces (it is unique upto an homeomorphism).

In the sequel we straightforwardly extend the relation "be a minor of" to the class of multigraphs (with the modification that after edge contraction we do not purge loops and multiple edges) and also to their drawings.

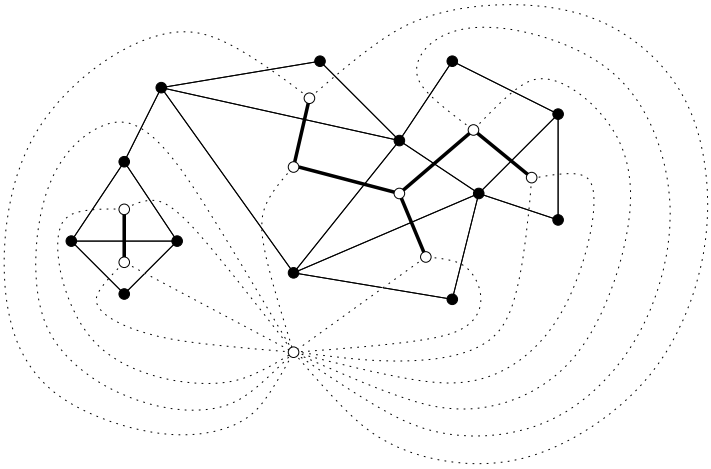


Figure 5: Example of an outerplanar drawing of G (black vertices), it's dual G^* (white vertices) and G^*_{in} (thick edges).

Observation 1.6. *If a drawing of H is a minor of a drawing of (multi)graph G then the drawing of H^* is a minor of the drawing of G^*.*

Proof. Without loss of generality we may assume that G and H have no isolated vertices, since they disappear in the construction of the dual.

- Edge removal in G corresponds to an edge contraction in G^* — two faces that were separated become a single face.
- After a contraction of an edge e in G the two adjoint faces have shortened their common boundary by the edge e — the edge e^* disappears in the dual.

□

Note that if H has no isolated vertices then in the above observation we can state an equivalence. This follows from the fact that $(H^*)^* = H$ in such a case.

We will continue with a special subclass of planar graphs. A planar graph is called *outerplanar* if it allows a drawing where all vertices lie on the outer face.

For a planar drawing of a graph G we denote G^*_{in} the subgraph of its dual induced by the inner faces. This multigraph represents the way the finite faces are connected together. and we will show that it can be done only in a tree-like manner, see Fig. 5.

Lemma 1.7. *A drawing G of a planar graph is outerplanar if and only if G^*_{in} is a forest.*

Proof. "⇒": any cycle, multiple edge or a loop in the dual separates at least one vertex of the original drawing from the outerface.

"⇐": If G^*_{in} is a forest, then the faces of G^* became as a partition of the only face of G^*_{in}. They all are incident with the only new vertex representing the outer face, hence in the original drawing of G the vertices they represent lie on the outer face as well. □

Theorem 1.8. *A graph G is outerplanar if and only if none of K_4 and $K_{2,3}$ is its minor.*

Proof. Firstly, K_4 and $K_{2,3}$ are two minor-minimal non-outerplanar graphs. As the class of outerplanar graphs is minor closed, K_4 and $K_{2,3}$ cannot appear as a minor of an outerplanar G.

For the opposite direction we show that every non-outerplanar graph G contains K_4 or $K_{2,3}$ as a minor. Without loss of generality assume that G is 2-connected, since each block of 2-connectivity can be treated separately.

We use the well known fact that every 2-connected graph can be inductively obtained from a cycle by adding edges and paths to prove a stronger claim: Every 2-connected graph with no K_4 and $K_{2,3}$ minor has an outeplanar drawing where the outer face is bounded by a convex polygon.

If G is a cycle, then the statement is clearly satisfied. Otherwise we identify and edge e or a path P, s.t. $G-e$ or $G-P$ is 2-connected[2]. By induction assume that $G-e$ or $G-P$ has a drawing according to the claim. In the first case, if e cannot be added to a drawing of $G-e$ then it means that e crosses another edge e'. Hence, the outer cycle together with e, e' yields a subdivision of K_4, a contradiction. In the other case, P can be added if and only if its ends are two consecutive vertices of the outer cycle of $G-P$. Otherwise P together with the outer cycle yield a subdivision of $K_{2,3}$. A simple geometric argument shows that P can be added such that the outer cycle of G is a convex polygon. □

Exercise 12: Prove Theorem 1.8 as a consequence of Kuratowski theorem.

2 Well quasi-orderings and the minor theorem

A binary relation $\preceq$ on a class X which is reflexive and transitive is called a *quasi-ordering*. If it is in addition antisymmetric, it is a *partial ordering*.

Exercise 13: Find all non-isomorphic partial orders and quasi-orders on a three-element set X.

Definition. A quasi-ordered class $(X, \preceq)$ is called well quasi-ordered, if for every infinite sequence $(a_i)_{i=1}^{\infty}$ of elements from X there exist elements a_i and a_j, such that $a_i \leq a_j$ and $i < j$.

Exercise 14: Are vectors of $\mathbb{N}^d$ or of $\mathbb{Q}_+^d$ well (quasi) ordered either with the order on all coordinates $\leq^d$ or with the lexicographic order $\leq^d_{\text{Lex}}$, respectively?

A pair a_i, a_j of a sequence $(a_i)_{i=1}^{\infty}$ in a quasi-ordered set $(X, \preceq)$ is called good, if $a_i \preceq a_j, i < j$. We call a sequence $a_1, a_2, \ldots$ bad, if it contains no good pair. Obviously well quasi-ordered sets are exactly those without bad sequences.

Observation 2.1. *In any well quasi-ordered class, every sequence $(a_i)_{i=1}^{\infty}$ contains a non-decreasing subsequence.*

[2]Here $G-P$ means the graph obtained by removal of the internal vertices of P.

Proof. The fact that a sequence is good implies directly that it contains infinitely many nondecreasingly ordered pairs. However, by use of the infinite Ramsey theorem, there must be also a nondecreasing subsequence: We construct an infinite graph on $\{a_i\}$, with edges connecting $(a_i, a_j) : a_i \preceq a_j, i < j$. Since it cannot contain an infinite independent set, it must contain an infinite clique, which corresponds to an infinite nondecreasing subsequence. □

We show that in a well ordered class, certain subclasses allow an easy description by forbidden elements.

Definition. We say that a set F is the *obstruction set* for a subclass Y of a quasi-ordered class X if the following duality holds:

$$\forall a \in X : \qquad a \notin Y \iff \exists b \in F : b \preceq a$$

Observation 2.2. *Let Y be a subclass of a well-ordered class X, such that Y is closed on smaller elemets in $\preceq$. (Formally: $b \preceq a, a \in Y \implies b \in Y$.) Then an obstruction set F exists, and is finite.*

Proof. Take all minimal elements of $X \setminus Y$ and pick a single representative from each equivalence class. This forms the set F.

As X is well ordered, there is no infinite set of incomparable elements. Hence, F is finite, since the elements of F are incomparable.

If $a \notin Y$ then a decreasing chain in $X \setminus Y$ starting from a must end in finitely many steps in an element of F, otherwise X would not be well ordered.

If $a \in Y$, then no $b \in F$ can satisfy $b \preceq a$, since as Y is closed under $\preceq$, it would give $b \in Y$, a contradiction. □

Observe that by the above construction the inclusion-wise minimal set F is unique under taking $\preceq$-equivalent elements.

Assume that a set X has a quasi-order $\preceq$. We extend this quasi-order onto the class of finite subsets of X denoted by $X^{<\omega}$ as follows: We say that finite sets $A, B \subset X$ satisfy $A \preceq B$, if there exists an injective mapping $f : A \to B$ such that $a \preceq f(a)$ for all $a \in A$.

Theorem 2.3 (Higman)**.** *If a set X is well quasi-ordered by $\preceq$, then $(X^{<\omega}, \preceq)$ is also a well quasi-ordered set.*

Proof. We prove the theorem by contradiction. Construct the sequence $(A_i)_{i=1}^{\infty}$ inductively as follows: We select A_i such that $|A_i|$ is minimal among all possible A_i such that an infinite bad sequence starts with $A_1, \ldots, A_i$. Such A_i exists for all $i \in \mathbb{N}$, and hence the sequence $(A_i)_{i=1}^{\infty}$ is well defined.

For each i we select some $a_i \in A_i$, and set $B_i = A_i \setminus \{a_i\}$. Since the set X itself is well quasi-ordered, there exists an infinite nondecreasing subsequence $a_{i_1}, a_{i_2}, \ldots$. The sequence $A_1, A_2, \ldots, A_{i_1-1}, B_{i_1}, B_{i_2}, \ldots$ contains a good pair, due to the minimality of selection of $A_{i_1} : |A_{i_1}| > |B_{i_1}|$.

The good pair of this sequence should not be of the form $(A_j, A_{j'})$, since $(A_i)_{i=1}^{\infty}$ is bad. Similarly the good pair is not of the form (A_j, B_{i_k}). In this case the relation $A_j \preceq B_{i_k} \preceq A_{i_k}$ yields the same contradiction.

However, the existence of a good pair of form $(B_{i_k}, B_{i'_k})$, causes a contradiction as well: Due to $a_{i_k} \preceq a_{i'_k}$ we get that $A_{i_k} \preceq A_{i'_k}$. □

Rado provided the following example showing that the Higman's theorem can not be extended to infinite countable sets.

First take $X = \mathbb{N}^2$ and a quasiorder $\preceq$ defined by $(a, b) \preceq (a', b')$ iff either $a = a'$ and $b \leq b'$ or alternatively $a, b \leq a'$. Observe that $(X, \preceq)$ is well quasi-ordered — every first coordinate may appear only finitely many times. Consequently the first coordinates are unbounded, so some a_i satisfies $a_i \geq a_1, b_1$.

In contrary $(X^\omega, \preceq)$ is not well quasi-ordered. Consider seqences $A_i = ((i, i+1), (i, i+2), \dots) \in X^\omega$. For any $i < j$ it holds that $A_i \not\succeq A_j$ easily. Simultaneously, $A_i \not\preceq A_j$, since $(i, j) \not\preceq (j, k)$ for any $(j, k) \in A_j$.

2.1 The minor order

Observe that the relation "H is a minor of G" is reflexive and transitive, in other words it defines a quasi-order on the class of finite graphs $\mathcal{G}$. For such situation we write $H \preceq G$.

Observe also that $\preceq$ is also antisymmetric when we restrict ourselves on non-isomorphic graphs.

Exercise 15: Find the minor ordering $\preceq$ on all non-isomorphic graphs on four vertices.

The following theorem is one of the jewels of graph theory:

Theorem 2.4 (Robertson and Seymour)**.** *The class of finite graphs ordered by the minor relation is well quasi-ordered.*

The complete proof of this theorem, formerly known as Wagner conjecture, has been published in a series of papers in Journal of Combinatorial Theory, Ser. B in late 80s. As its proof is far behind the scope of this course we present much simpler variant of the "Excluded Minor Theorem", namely for the class of forests. We further discuss some ideas which were used in the general proof.

The most important corollary of Theorem 2.4 is that every minor-closed class of graphs can be characterized by a finite obstruction set, in this context often called *Kuratowski set*.

Also observe that this gives us a positive answer to the question whether Kuratowski theorem can be generalized for surfaces of higher genus, since such graph class is minor-closed. However, the number of forbidden minors grows fast: graphs embedable into the projective plane have 35 forbidden minors (surprisingly, 3 of them are disconnected) [3]. For no other surface than the Euclidean plane and the projective plane the obstruction set is known.

The test "Is H a minor of G?" can be done in time $O(|V_G|^3)$ for a fixed graph H [35].

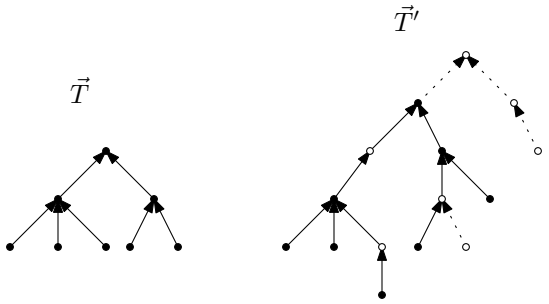


Figure 6: The topological minor relation.

Corollary 2.5. *The membership test for a minor closed class of graphs can be performed in $O(n^3)$ time. (The constant hidden in the O-notation depends on the graph class.)*

In particular, any class of graphs embedable on a fixed surface can be recognized even faster [24].

Theorem 2.6 (Mohar)**.** *For every surface S there exists a linear-lime algorithm that decides whether a given graph G can be embedded into S.*

Due to Theorem 2.3 it suffices to show well quasi-ordering on connected graphs (which usually do not form a minor-closed set), and the well quasi-order will be also transferred to finite disjoint unions of such graphs. (E.g., from trees to forests.)

Now we are ready to prove the minor theorem for the class of forests. We prove a stronger version of the theorem (with an easier proof): Consider only rooted trees and only those minor relations that obey the parent-child relation (also in the sense of Lemma 1.1). More precisely we define:

Every rooted tree T has a unique orientation $\vec{T}$ such that each edge is oriented towards the parent vertex.

We extend the notion of being a topological minor to rooted trees in a the following sense: A rooted tree T is a topological minor of a rooted tree T', if $\vec{T}$ can be obtained from a subtree of $\vec{T}'$ by a series of contractions of edges incident with vertices of indegree one (see Fig. 6).

Theorem 2.7 (Kruskal)**.** *The class of finite rooted trees is well quasi-ordered by the topological minor relation.*

Proof. As in the previous proof, we will proceed by contradiction. First, we construct a bad sequence of rooted trees $(T_i)_{i=1}^{\infty}$, such that T_i is selected as the minimal rooted tree such that $T_1, \ldots, T_i$ has an infinite bad extension.

Denote by r_i the root of tree T_i, and let F_i be the forest of rooted trees $T_i - r_i$, where the children of r_i are selected as roots of the new trees.

It is enough to prove that the set $\mathcal{F} = \bigcup_{i=1}^{\infty} F_i$ is well quasi-ordered by the topological minor relation. Consider any sequence $(B_j)_{j=1}^{\infty}$ of trees from $\mathcal{F}$, and for each j select arbitrarily $i(j)$, such that $B_j \in F_{i(j)}$. Select a subsequence $B_{j_1}, B_{j_2}, \ldots$ such that $i(j_1), i(j_2), \ldots$ is non-decreasing.

The sequence $T_1, T_2, \ldots, T_{i(j_1)-1}, B_{j_1}, B_{j_2}, \ldots$ is not bad due to the minimal selection of $T_{i(j_1)}$, hence it contains a good pair, either of form (T_i, B_j) or $(B_j, B_{j'})$. Any good pair of the first form satisfies $T_i \preceq B_j \preceq T_{i(j)}$ and $i < i(j_1) \leq i(j)$. The existence of such pair contradicts the assumption that the sequence $(T_i)_{i=1}^{\infty}$ is bad. A good pair of the second form is also a good pair for the sequence $(B_j)_{j=1}^{\infty}$. Hence, the set $\mathcal{F}$ is well quasi-ordered.

Finally, we apply Theorem 2.3 and get that $\mathcal{F}^{<\omega}$ is well quasi-ordered too. Therefore, any extension of the topological minor relation $F_i \preceq F_{i'}$ can be extended to a minor relation $T_i \preceq T_{i'}$. Observe that to get a minor of T_i in $T_{i'}$, it is enough to contract paths in $T_{i'}$ to the roots of minors in $F_{i'}$ into a single edge. □

The proof of Robertson-Seymour theorem follows in some situations the above proof of Kruskal theorem. The central notion on the general proof takes the notion of treewidth. Graphs with bounded treewidth have a tree-like structure, hence similar arguments might be used. We introduce this notion in the next section.

Exercise 16: Show that the subgraph relation is not well-ordering of all graphs. I.e., exhibit a class of graphs that is closed on taking subgraphs which has infinitely many forbidden subgraphs.

Exercise 17: Find the obstruction set for the class of all forests and for the class of graphs of maximum degree two. What is the intersection of these two classes?

Exercise 18: Show that the finite trees are not well quasi-ordered by the subgraph relation.

3 Treewidth

Treewidth was used first by Halin in 1976 in attempt to solve the Hadwiger conjecture [16]. In 1980 Proskurowski rediscovered the concept of k-trees for its algorithmic consequences [29]. Independently, Robertson and Seymour used tree decompositions as a main tool in their proof of the Wagner's conjecture [33].

3.1 Definition and basic properties

We start with the formal definition of tree-decomposition and treewidth.

Definition. The *tree decomposition* of a graph G is a tree T satisfying the following properties:

- $V(T) \subseteq \mathcal{P}(V_G)$, i.e., the nodes of T are subsets of V_G.
- For each edge $(u, v) \in E_G$ there is a node X_i of T, such that both $u, v \in X_i$.
- For each vertex $u \in V_G$, the subgraph of T induced by sets containing u is nonempty and connected (i.e. it is a subtree).

The width of a decomposition T is $\max_{X_i \in V_T} |X_i| - 1$. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width among all its decompositions.

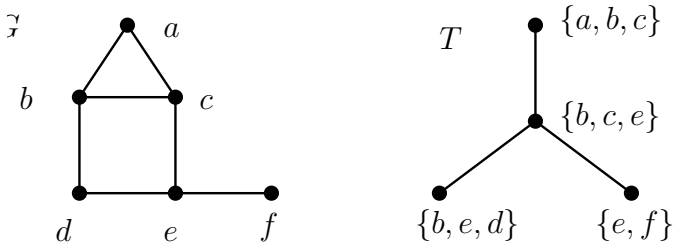


Figure 7: An example of a tree decomposition of a graph.

In the following text we call the vertices of the tree T *nodes*, to distinguish them from the vertices of the graph G.

Exercise 19: Prove that the connectivity condition in the last item of the definition of the tree decompositionis the equivalent with: "If a node X_j of T lies on the unique path between nodes X_i and X_k then $X_i \cap X_k \subseteq X_j$".

Observation 3.1. *The treewidth of the disjoint union* $G_1 \cup G_2$ *is* $\max\{\text{tw}(G_1), \text{tw}(G_2)\}$.

We show that vertices of each complete subgraph of G form a subset of some node in any tree decomposition of G. This yields the following lemma:

Lemma 3.2. *For every graph* G *it holds that* $\text{tw}(G) \geq \omega(G) - 1$.

Exercise 20: Prove Lemma 3.2

Proposition 3.3. *If* H *is a minor of* G*, then* $\text{tw}(H) \leq \text{tw}(G)$.

Proof. If H is a subgraph of G, then $\text{tw}(H) \leq \text{tw}(G)$, because from a minimal tree-decomposition of G we delete all vertices outside H and get a tree decomposition of H of width at most $\text{tw}(G)$. (On edge removal no modification of the tree decomposition is necessary.)

So the only non-trivial part corresponds to the edge contraction. Assume inductively that H is a graph that came out of G after contracting an edge (u, u') into the new vertex v. Then in the minimal tree decomposition replace each occurrence of u or u' by v and get a tree decomposition of H. □

Proposition 3.4. *If* G *is a subdivision of* H*, then* $\text{tw}(H) = \text{tw}(G)$.

Proof. If $\text{tw}(H) = 0$ or 1 then the claim is valid, since in an edgeless graph there is no edge to be subdivided (the first case) and the subdivision of a forest is a forest (the second case).

Assume now that T is a tree decomposition of H of width $\text{tw}(H)$. Let $G = H \cdot (u, v)$ and v' be the new vertex of G. Then T has a node X_i containing both u and v. By attaching a new node $\{u, v, v'\}$ to X_i we obtain a decomposition of G of the same width. Consequently $\text{tw}(G) \leq \text{tw}(H)$

As H is a minor of G we get by Proposition 3.3 that $\text{tw}(H) = \text{tw}(G)$. For a series of individual subdivisions we apply induction. □

Exercise 21: Determine the treewidth of a tree (or of a forest).

Exercise 22: Determine the treewidth of the cube Q_3.

Exercise 23: Determine the treewidth of a complete bipartite graph.

Exercise 24: What is the treewidth of a cycle, or of an outerplanar graph, respectively?

Exercise 25: Find a graph of treewidth 2 that is not outerplanar.

3.2 Recursively defined graph classes

Definition. A *k-terminal* graph is a graph with k distinguished vertices, $t^1, \ldots, t^k$ called *terminals*.

Definition. We say that $\odot$ is an m-ary graph *connecting operation* on k-terminal graphs, if for $H = \odot(G_1, \ldots, G_m)$ holds that H is constructed from the disjoint union of $G_1, \ldots, G_m$ by further unifications prescribed by $\odot$. Within each unification from each G_i is chosen either a predetermined terminal or no vertex and the vertices of the resulting set (which must have at least one vertex) are merged together. Each terminal of each G_i participates in a unique unification. It is also specified, whether such newly constructed vertex is a particular terminal of H or whether it looses its terminal status.

Formally, the operation $\odot$ can be described as a partition of the union of the sets of terminals of $G_1, \ldots, G_m$, together with a bijection between the set of terminals $t^1, \ldots, t^k$ of H and chosen k sets of the partition. It is required that no set of the partition contains two terminals from the same graph G_i and that all these sets are nonempty.

For example consider the following binary connection operations on 2-terminal graphs:

- the *serial connection* $\odot_s$: identify t_1^2 with t_2^1; the new terminals are $t^1 = t_1^1$ and $t^2 = t_2^2$,
- the *parallel connection* $\odot_p$: merge t_1^1 with t_2^1 into the new terminal t^1, and analogously $\{t_1^2, t_2^2\} \to t^2$,
- the *jackknife connection* $\odot_j$: let $t^1 = t_1^1$, and also merge t_1^2 with t_2^1 into t^2.

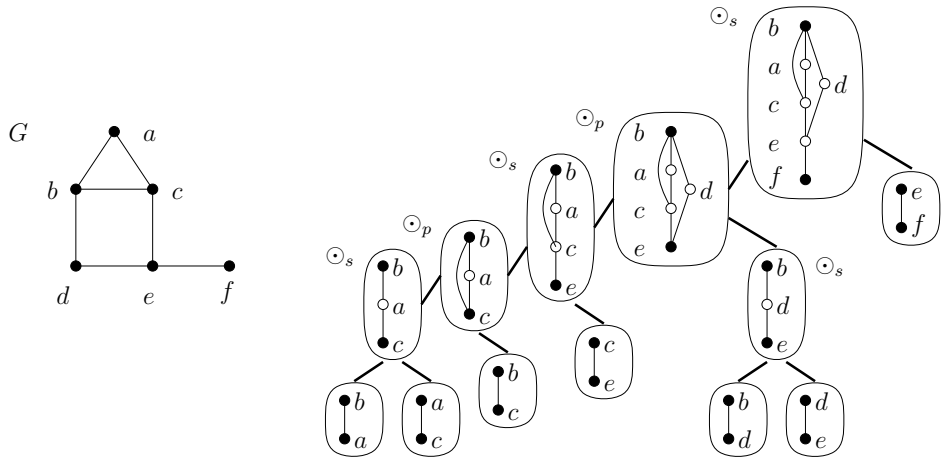
Definition. Recursively defined graph class given by a pair (O, B) contains all graphs that can be constructed from graphs from the base set B by connecting operations from the set O.

The membership of any graph from such set can be justified by so called construction term or construction tree.

Observation 3.5. *The recursive class given by $(\{\odot_j\}, \{K_2\})$ is the class of all trees.*

Definition. The class given by $(\{\odot_s, \odot_p\}, \{K_2\})$ is the class of *series-parallel graphs*.

See Fig. 8 for an example a series-parallel graph together with its construction tree and construction term.



$$G = ((((K_2 \odot_s K_2) \odot_p K_2) \odot_s K_2) \odot_p (K_2 \odot_s K_2)) \odot_s K_2$$

Figure 8: An example of the construction tree and construction term of a series-parallel graph. In each graph used in the recursion the two terminals are colored black.

Theorem 3.6. *A graph is of treewidth at most two if and only if it is a subgraph of a series-parallel graph.*

Proof. We first prove by induction that the treewidth of a series-parallel graph is bounded by two. We use decompositions where the two terminals t^1 and t^2 belong to one of the nodes. Then to implement the parallel operation we join the two trees by an edge between the two special nodes. The serial operation can be implemented by connecting the two fragments by a node consisting of all three terminals $t_2^1, t_1^2 = t_2^1$ and t_2^2.

Now consider a graph G of treewidth at most two. Without loss of generality we may assume that it is connected. Any tree decomposition of G with nodes of size three can be modified by insertion of new nodes such that in the final decomposition T every two adjacent nodes intersect on exactly two vertices of G.

Let G' be the graph which arose by adding edges of (u, v) into G whenever u and v belong to the same node. We show that G' is series-parallel.

Any leaf node of T contains a vertex u such that u does not appear elsewhere in T. The graph $G' \setminus u$ is of treewidth 2 and is series-parallel by the induction hypothesis. Let v, w be the two vertices which appear in the same node as u. Since (v, w) is an edge of $G' \setminus u$, it had to be inserted as an edge K_2 in its construction tree. But, at this moment we can extend K_2 onto the triangle K_3 by a parallel connection with the path vuw and get the construction tree for G'. □

Theorem 3.7. *A graph is a subgraph of series-parallel if and only if it has no K_4 minor.*

Proof. First observe that every 3-connected graph contains a K_4 minor: Choose any pair of vertices u, v. By 3-connectivity we can find three internally disjoint paths P_1, P_2, P_3 between u and v, and at lest two of them, say P_2 and P_3 have length at least two. Moreover

$G \setminus \{u, v\}$ is also connected, in particular one can identify the shortest path path P_4 between two internal vertices of P_2 and P_3. These paths $P_1, \ldots, P_4$ yield a K_4 subdivision.

If G is not connected, we insert new edges between components to make it connected. Every added edge is a bridge, hence it cannot form a K_4 minor.

If u is an articulation and v, v' it neighbors from distinct 2-connected blocks, we add the edge (v, v'). This operation avoids creation of a K_4 minor: Vertices u, v form a cutset, hence they could only belong to the same path of a K_4 subdivision. As they are are adjacent, they could be chosen consecutive on the path. Afterwards the K_4 minor would avoid any edge of the 2-connected block containing v'.

When G has at least 4 vertices, it has a 2-cut. Choose a cutset $\{v, w\}$ so that $G \setminus \{v, w\}$ has a component on the smallest possible subset of vertices U. Consider the graph H formed from the subgraph of G induced by $U \cup \{v, w\}$ together with the edge (v, w), if it was not present in the subgraph. By the same argument as above H has no K_4 minor, as otherwise it would appear already in G.

Moreover H is 2-connected and every 2-cut of H would be a 2-cut of G separating even smaller subset of vertices. Thus $H \simeq C_3$, and U consists of a single vertex u. If v and w are not adjacent in G we can add the edge (v, w) without forming a K_4 minor.

Then, as in the proof of Theorem 3.6 we apply induction on $G' = G \cup (v, w) \setminus u$ to obtain a construction tree for (a supergraph of) G. □

Series-parallel graphs are often defined as K_4 minor free graphs or alternatively with help of ≤ 2-sums. Here G is k-sum of a graphs G_1 and G_2 if it can be formed from $G_1 \cup G_2$ by one by one merge of vertices of a k-clique in G_1 with vertices of a k-clique in G_2 followed by possible deletion of edges in the resulting k-clique.

Observe that from Lemma 1.7 follows the treewidth of a dual of an outerplanar graph is at most two. Robertson and Seymour conjectured in 1986 [34] that the treewidth of a planar graph differs from th treewidth of its dual by at most one. This conjecture was proved in 1996 by Lapoire [21], see a paper by Bouchitté et al. for a simpler proof [8].

3.3 Chordal graphs

For a better understanding of the structure of graphs of bounded treewidth, we give another characterization in terms of k-trees.

Definition. A graph is called *chordal*, if it contains no induced cycle of length at least four as a subgraph.

We note here that chordal graphs have nice properties from the algorithmic point of view. As they are perfect, they allow a polynomial time algorithms for several problems that are NP-hard for general graphs, like the coloring problem. In particular we later show a similar coloring algorithm for graphs of bounded treewidth.

Observe that chordal graphs can be obtained from a complete graph by repeating the following operation: Find a set of mutually adjacent vertices and add a new vertex adjacent to all of them.

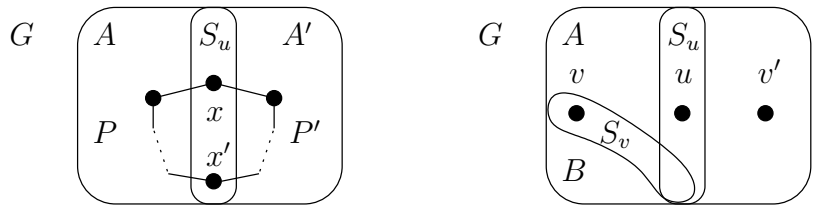


Figure 9: Two illustrations for the proof of Proposition 3.8.

Proposition 3.8. *Every chordal graph G has a vertex whose neighborhood induces a complete subgraph in G.*

Proof. For a contradiction assume that this is not the case and take a graph G with this property. Then any vertex u has at least one pair of neighbors v and v' such that $(v, v') \notin E_G$.

Now choose a minimal set of vertices S_u that v and v' belong to different components A and A' of $G \setminus S_u$. We claim that such S_u induces a clique in G. For contrary assume that for some $x, x' \in S_u$ are nonadjacent. Then take two shortest paths P and P' between x and x' whose internal vertices are in A and A' resp. As both x, x' are adjacent to both A, A' (otherwise S_u is not minimal) these two paths must exist. But their composition yields a chordless cycle of length at least 4, a contradiction.

Denote $A_1, \ldots, A_k$ the components of $G \setminus S_u$. Clearly, any other S_v must lie in one of $A_i \cup S_u$. Such S_v may not have a nontrivial intersection with two components A_i, A_j because the two components would become adjacent in $G \setminus S_u$. We call this property that the sets S_u and S_v are *noncrossing*.

Select u such that $G \setminus S_u$ contains a component A of the smallest size and take $v \in A$. As S_u and S_v are noncrossing, at least one component B of $G \setminus S_v$, satisfy $B \subsetneq A$, a contradiction on the minimality of $|A|$ (see Fig. 9). □

Note that sets S_u are special kinds of separators. Importance of separators for bounded treewidth graphs will be in detail discussed in section 3.5.

The property of Proposition 3.8 can be used iteratively and produce so called *perfect vertex elimination scheme* of a graph. It is an ordering of it's vertices $v_1, \ldots, v_n$ such that for any vertex v_i its higher numbered neighbors $\{v_j : j > i, v_j \in N(v_i)\}$ induce a clique in G.

Corollary 3.9. *Every chordal graph has a tree decomposition such that every node induces a clique.*

Proof. The insertion of a vertex u, whose neighborhood induces a clique, corresponds with an insertion of a leaf node on $\{u\} \cup N(u)$ which also forms a clique. □

Definition. A graph is called k-tree, if it can be constructed from the complete graph K_{k+1} by a sequence of vertex insertions. Each time a new vertex is inserted, it is made adjacent to some clique of size k.

Observe that k-tress are chordal. The same graph class can be obtained by generalized jackknife operations on $(k+1)$-terminal graphs. Here in $G_1 \odot_j G_2$ some k terminals of G_2 are merged with terminals of G_1, and terminals of the result are the same as in G_1. It is necessary to allow all $k+1$ possible jackknife oprerations.

In another point of view a k-tree is a graph G which has a perfect vertex elimination scheme where each vertex v_i has exactly $\min\{k, |V_G| - i\}$ following neighbors.

Due to the following fact, the graphs of treewidth at most k are also sometimes called *partial k-trees*.

Lemma 3.10. *Every graph of treewidth at most k is a subgraph of a k-tree and vice versa.*

Proof. First, if a node X_i has size at most k insert into X_i a vertex from a neighbouring node as long as all nodes are of size $k+1$. Then add extra nodes such that adjacent nodes have k common vertices. These manipulations do not affect the width of the decomposition.

Observe that the insertion of edges that have both ends in the same node does not increase the treewidth of the graph, but yields to a k-tree. (The construction of the k-tree can be derived from the tree decomposition.)

For the opposite implication it is easy to show that every k-tree has treewidth at most k. The tree decomposition of width k can be derived from the construction of the graph. □

Corollary 3.11. *Show that* $\mathrm{tw}(G) = \min\{\omega(G') : G \subseteq G', G' \text{ is chordal}\} - 1$

Corollary 3.12. *Every graph G has an optimal tree decomposition on at most $|V_G|$ nodes.*

Exercise 26: Show that chordal graphs are intersection graphs of subtrees of a tree.

Exercise 27: Show that graphs G of treewidth at most k with $k \geq 1$ have strictly less than $k|V_G|$ edges.

Exercise 28: Design a polynomial-time coloring algorithm for chordal graphs.

3.4 Search games

We relax from the formal presentation and present a simple "cops-and-robber" game (by LaPaugh [20]) played on a graph. Imagine that the graph represents a town with vertices as crossings and edges as streets. The game is played between one robber on a motorbike and several police helicopters. The police always knows the position of the robber, but the robber on a motorbike can quickly move from a vertex to any of its neighbors. The police helicopters do not need to follow edges of the graph, however their movement needs much more time. The robber may move through several edges during a single movement of a police helicopter, however he cannot pass through a crossing occupied by a helicopter. Even if a helicopter decides to land at the robber's standpoint, the robber has enough time to run away through any free adjacent vertex. The robber's aim is to escape forever, whereas the police wants to encircle him, forbid any his movement and finally capture him by landing a helicopter on the vertex of his position.

The classification winning strategies for the game is given by the following proposition:

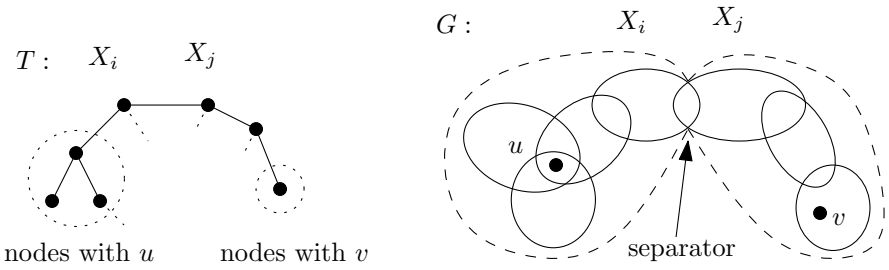


Figure 10: An illustration for Proposition 3.14.

Proposition 3.13. *The police has a winning strategy with $k+1$ helicopters if and only if the graph has treewidth at most k.*

We partially prove the proposition in the forthcomming exercises. Try to find some property of a graph that provides a winning strategy for the robber. Such a property would yield better lower bound on a treewidth of a graph.

Exercise 29: Assume that the police has two helicopters. On which graphs the robber has a winning strategy, and on which the police wins?

Exercise 30: Prove the $\Leftarrow$ implication of Proposition 3.13.

Exercise 31: Decide who wins on the $n \times n$ grid, depending on the number of helicopters. What is the consequence for the treewidth of a planar graph? (Assume correcteness of Proposition 3.13.)

3.5 Separators

Definition. If V_G can be partitioned into three disjoint sets A, B and S such that every path from a vertex from A to a vertex from B contains a vertex from S, then we say that S *separates* A and B, or, shortly, that S is a *separator*.

By the definition, S may be empty if A and B are formed from different components of G.

Separators or also called cutsets, play important role in the finding good tree decompositions. We show that nodes of a tree decomposition are good candidates for separators.

Proposition 3.14. *Let T be a tree decomposition of a connected graph G. For every edge $(X_i, X_j) \in E_T$ the following holds:*

- *either the intersection $X_i \cap X_j$ is a separator in G*
- *or all nodes of the component of $T - (X_i, X_j)$ containing the node X_i are subsets of X_j.*
- *or the vice-versa for X_j*

Proof. Let T be a tree-decomposition of G. Assume X_i and X_j are two nodes of T that violate the last two conditions of the lemma. Hence a vertex u exists such that $u \in X_{i'} \setminus X_j$ where $X_{i'}$ is behind X_i (i.e., u belongs to the component of $T - (X_i, X_j)$ containing X_i). Also we can find $v \in X_{j'} \setminus X_i$ where $X_{j'}$ is behind X_j.

Assume for a contradiction that u and v are connected by a path in $G \setminus (X_i \cap X_j)$. See Fig. 10.

Vertices of a path from u to v induce in T a subtree, hence the tree must contain the edge (X_i, X_j). Thus a vertex from this path belongs to $X_i \cap X_j$ and u and v are separated in $G \setminus (X_i \cap X_j)$, a contradiction. □

Definition. For a graph G and a set of vertices $W \subseteq V_G$ we say that S *separates* W, if at least two vertices of W are separated by S, i.e. such S separates A and B where both intersect W.

We say that a separator S is *good* for a given set W, if S separates sets A and B, where both contain at least one and at most $\frac{2}{3}|W|$ vertices of W.

Note that neither A nor B have to induce connected subgraphs of G.

Lemma 3.15. *If the treewidth of a graph G is at most k, then any $W \subseteq V_G$ of size at least $2k + 3$ has a good separator S of size at most $k + 1$.*

Proof. Take a tree-decomposition T of G of width k, where each node has at most three neighbors. We will show that some node X_i of T is a good separator for W.

If some node X_i has the property that some of the at most three subtrees of $T \setminus X_i$ contains more than $\frac{1}{2}|W|$ vertices of $W \setminus X_i$, then orient the corresponding edge from X_i. In particular, each edge adjacent to a leaf X_i is oriented from X_i, because $|W \setminus X_i| \geq 2k + 3 - (k + 1) = k + 2 > \frac{1}{2}|W|$.

No edge may be oriented in both directions, as on both sides would be more than $\frac{1}{2}|W|$ vertices of W. Hence there exists an internal node X_i with no outgoing edge.

If the internal node X_i is of degree two, we take $S = X_i$.

Otherwise the three subtrees of $T \setminus X_i$ provide a partition of $W \setminus X_i$ into three subsets, W_1, W_2 and W_3, each of size at most $\frac{1}{2}|W|$. Without loss of generality assume that $|W_1| \geq |W_2| \geq |W_3|$.

Now assume for a contradiction that $|W_2 \cup W_3| > \frac{2}{3}|W|$. Consequently, $|W_2| > \frac{1}{3}|W|$ as well as $|W_1| < \frac{1}{3}|W|$, which contradicts the choice $|W_1| \geq |W_2|$. Hence $S = X_i$ is a good separator, if A and B are chosen such that $W_1 \subseteq A$ and $W_2 \cup W_3 \subseteq B$. □

Indeed, a slightly stronger variant of the last argument of the proof will be useful later. We state it separately as follows:

Observation 3.16. *If W_1, W_2, W_3 are disjoint subsets of W, such that none contains more than $\frac{2}{3}|W|$ elements of W, then for some pair W_i, W_j it holds that the union $W_i \cup W_j$ also does not contain more than $\frac{2}{3}|W|$ elements of W.*

In Section 5.1 we will show that the existence of good separators implies bounded treewidth:

Theorem 3.17 (Reed [30]). *If every set W of size at least $2k+3$ has a good separator of size at most $k+1$ then $\mathrm{tw}(G) \leq 4k+3$.*

Sets without good separators hence provide arguments for showing high treewidth.

Definition. A set $W^* \subseteq V_G$ is *strongly k-linked* if it has no good separator with at most k vertices.

Corollary 3.18. *If G has treewidth at least $4k$ then it contains a strongly k-linked set.*

Some classes of graphs allow separators of sufficiently small size. For instance, planar graphs on at least five vertices have separators of size at most five, since they always have a vertex of degree at most five. Now we show that they contain also small good separators for $W = V_G$.

Theorem 3.19 (Lipton-Tarjan [23]). *Every planar graph G on n vertices has a separator S of size at most $2\sqrt{2n}$ such that each component of $G \setminus S$ has less than $\frac{2}{3}n$ vertices.*

Proof (Alon, Seymour, Thomas [2]). Without loss of generality assume that G is triangulated. Sets A, B, C form a partition of the vertex set of G such that C is a cycle separating A and B. The set A consists of the vertices inside C, and B outside.

Denote $k := \lfloor\sqrt{2n}\rfloor$. We select C such that $|C| \leq 2k$, $|B| < \frac{2}{3}n$ and $|A| - |B|$ is as small as possible. Assume for a contrary that $|A| \geq \frac{2}{3}n$.

For $u, v \in C$ let $c(u,v)$ be the distance between u and v along the cycle C, and $d(u,v)$ their distance measured in the graph D encircled by the cycle C (consisting of the cycle edges, edges between C and A, and of the subgraph induced by A).

We claim that for any $u, v \in C$ the equality $c(u,v) = d(u,v)$ holds. We prove this claim by contradiction. Take u, v such that $c(u,v) > d(u,v)$ and $d(u,v)$ is minimal. Let P be the path of length $d(u,v)$ between u and v. This P splits the set A into two pieces A_1 and A_2. Assume that $|A_1| \geq |A_2|$ and further denote A_1, B_1 and C_1 to be the partition of G derived from A_1. We show that this partition violates the optimal choice of A, B and C. Observe that

$$n - |B_1| = |A_1| + |C_1| \geq \frac{1}{2}\left(|A_1| + |A_2| + |P| - 2\right) = \frac{1}{2}|A| \geq \frac{1}{3}n.$$

Consequently, $|B_1| < \frac{2}{3}n$. Since P is a shortcut of C we have $|C_1| \leq |C| \leq 2k$. Finally, as A_1 is a subset of A, the difference $|A_1| - |B_1|$ is smaller.

We continue with the main thread of the proof. Without loss of generality we assume that $|C| = 2k$, (if C would be smaller we extend it by some triangles inside A and get a better partition). Let $v_0, v_1, \ldots, v_{2k} = v_0$ be the vertices of C.

We claim that D has $k+1$ disjoint paths starting from $v_0, \ldots, v_k$ and ending in $v_{2k}, \ldots, v_k$ (the first and the last path has length zero). If these paths would not exist, then by Menger's theorem D has a cut of size k. This cut also yields a path of length $k-1$ between v_0 and v_k, a contradiction.

The total number of vertices of these paths is at least:

- $1+3+5+\cdots+k+k+\cdots+1 = 2\left(\frac{k+1}{2}\right)^2 = \frac{1}{2}(k+1)^2$ for k odd,
- $1+\cdots+(k-1)+(k+1)+(k-1)+\cdots+1 = k+1+2\left(\frac{k}{2}\right)^2 > \frac{1}{2}(k+1)^2$ for k even.

In both cases we derive that $|V_D| \geq \frac{1}{2}(k+1)^2 = \frac{1}{2}\left(\lfloor\sqrt{2n}\rfloor+1\right)^2 > n$, a contradiction. □

We have already mentioned in Section 3.4 that planar $n \times n$ grid G has treewidth n, i.e., $\mathrm{tw}(G) = \sqrt{|V_G|}$. In particular, these grids are planar graphs with asymptotically maximal treewidth.

Exercise 32: Show that for each $W \subseteq V_G$, and any tree decomposition T of G it holds that either W is a subset of some node $X_i \in V_T$ or W is separated by $X_i \cap X_j$ for some edge $(X_i, X_j) \in E_T$.

Exercise 33: For each k construct a graph G of treewidth k and a set W s.t. any separator S of size k separates two *connected* components A and B, where one of these contains $\frac{k}{k+1}|W|$ vertices of W.

(Hence Lemma 3.15 is tight in the sense that for $k \geq 3$ we cannot force at most $\frac{2}{3}|W|$ vertices of W in both parts A and B with cuts of size k.)

Exercise 34: Show that $tw(G) = O\left(\sqrt{|V_G|}\right)$ for all planar graphs G.

3.6 Brambles

We now focus our attention on a method to prove a lower bound on treewidth. For this purpose we define a notion of bramble:

Definition. A family $\mathcal{B}$ of subsets of vertices of G is called *a bramble*, if

- Each $B \in \mathcal{B}$ induces a connected subgraph of G.
- For each pair of sets B, B' in $\mathcal{B}$ the union $B \cup B'$ induces a connected subgraph in G. In other words either B and B' have at least one vertex in common, or there exist an edge between B and B'.

We say that a set W covers a bramble $\mathcal{B}$ if the set W intersects every set B of the bramble $\mathcal{B}$. The *order* of a bramble $\mathcal{B}$ is the minimum number of vertices of a set that covers $\mathcal{B}$.

Exercise 35: Find a bramble of order $n+1$ in the $n \times n$ grid graph.

Lemma 3.20. *If G has a bramble $\mathcal{B}$ of order $k+1$, then* $\mathrm{tw}(G) \geq k$.

Proof. Let T be a tree-decomposition of G. We claim that there exists a node of T that intersects all sets in the bramble $\mathcal{B}$.

If not, then for every node X_i at least one $B \in \mathcal{B}$ exists such that X_i does not intersect B. As B is connected, the nodes containing vertices of B form a connected subtree T' of

T. This subtree T' does not contain X_i, hence we identify the unique edge incident with X_i separating X_i from T'. We orient this edge towards T'.

Now, some edge (X_i, X_j) is oriented in both directions. The cutset $X_i \cap X_j$ separates sets B_i and B_j, a contradiction. □

In fact, the opposite implication in Lemma 3.20 holds, but it is not trivial to prove. The theorem is called the *treewidth minimax theorem* and was proved by Seymour and Thomas in 1993 [37].

Theorem 3.21. *G contains a bramble of order $k+1$* if and only if $k \leq \mathrm{tw}(G)$.

E.g., the $n \times n$ grid has brambles of order $1, \ldots, n+1$, while it is of treewidth n.

In the usual notation this theorem can also be written as: $\max_{\mathcal{B}} \min_{W} |W| = \min_{T} \max_{X} |X|$

The forward implication of Theorem 3.21 has been already proved by Lemma 3.20. The backward implication follows from the following stronger theorem, which is better suited to be proved by induction:

Theorem 3.22. *Let G have no bramble of order $k+1$. Then for each bramble $\mathcal{B}$ there exists a tree decomposition T, such that each node $X \in V_T$ of size at least $k+1$ is a leaf in T and this node also does not cover $\mathcal{B}$.*

Proof of Theorem 3.21. By contrary assume that G has no bramble of order $k+1$. Then apply Theorem 3.22 on the bramle $\mathcal{B} = \{V_G\}$. This bramble is covered by any nonempty set. Hence the corresponding tree decomposition T may not contain a node of size $k+1$, therefore $tw(G) < k$. □

Proof of Theorem 3.22. We use induction on the brambles, ordered by the number of covering sets of size at most k. For any bramble $\mathcal{B}$ assume that the statement is valid for all brambles which have smaller number of small covering sets.

Without loss of generality we may assume that $|V_G| \geq k+1$ since otherwise we can take T as the one-vertex tree with $X = V_G$.

For a given bramble $\mathcal{B}$ select the set W of the minimal size that covers $\mathcal{B}$. Since no bramble has order at least $k+1$, we have $|W| \leq k$. Denote by $A_1, \ldots A_r$ the components of the graph $G \setminus W$. Since $|V_G| \geq k+1$ we have that at least one such component exists.

We now prove an auxiliary statement:

For any $i = 1, \ldots, r$ there exists a tree decomposition T_i of $G_i = G|_{A_i \cup W}$ such that:

- *W is a node of T_i,*
- *each node of size at least $k+1$ in T_i is a leaf and does not cover $\mathcal{B}$.*

We distinguish two cases. Firstly, if $\mathcal{B} \cup \{A_i\}$ is not a bramble then T_i is the tree on two nodes W and $A_i \cup N(A_i)$. Since A_i is separated from some $B \in \mathcal{B}$, we have $B \cap (A_i \cup N(A_i)) = \emptyset$, i.e., $A_i \cup N(A_i)$ does not cover $\mathcal{B}$.

For the other case we assume that $\mathcal{B}' = \mathcal{B} \cup \{A_i\}$ is a bramble. Every set that covers $\mathcal{B}'$ covers also $\mathcal{B}$. In addition, the set W covers $\mathcal{B}$ but not $\mathcal{B}'$ as W is disjoint with A_i. Then $\mathcal{B}'$

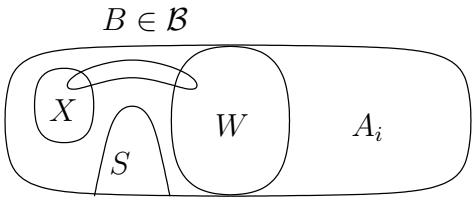


Figure 11: No small separation exists between X and W.

has smaller number of covering sets of size at most k than $\mathcal{B}$. By the induction hypothesis there is a tree decomposition T' of G for $\mathcal{B}'$ with properties given by the statement of the theorem.

If T' is valid also for $\mathcal{B}$ no more construction is needed. Otherwise T' contains a node X of size at least $k+1$, which is a leaf disjiont with A_i and which covers $\mathcal{B}$.

Observe that no set S of size less than $|W|$ separates W and X, since it would had to cover $\mathcal{B}$ (see Fig. 11). By Menger's theorem there exists a set of $|W|$ vertex-disjoint paths which connect W and X. Without loss of generality we may assume that each such path intersects W only in one vertex.

We now modify the tree decomposition T' into T_i as follows:

1. Restrict all nodes of T' to $A_i \cup W$.
2. For every $w \in W$ insert w to all nodes that are in T' on a path between X and a node with w.

Observe that except the node X all nodes with added w are internal in T'. Thus each leaf node of T_i is a subset of the original leaf node of T' (except X). Also the size of no node has increased, since the addition of w is compensated by the deletion of some vertex from the path between W and X starting from w. Any node Y of T_i of size at least $k+1$ was of size at least $k+1$ already in T'. By the statement of the theorem, Y is a leaf and does not cover $\mathcal{B}' = \mathcal{B} \cup \{A_i\}$. As $Y \subset W \cup A_i$ and W has at most k vertices, the node Y contains at least one vertex of A_i. Therefore Y is disjoint with some $B \in \mathcal{B}$, hence it does not cover $\mathcal{B}$.

In the final step in the construction of T (if it wasn't found earlier) we take a disjoint union of the auxiliary tree decompositions $T_1, \ldots, T_r$ and merge the nodes with W into a single node. □

Exercise 36: Show that the existence of a bramble of order $k+1$ gives a robber a winning strategy if the police has at most k helicopters.

(Together with Theorem 3.21 this proves the $\Rightarrow$ implication of Proposition 3.13.)

3.7 Treewidth and the minor theorem

In this section we overview main ideas that were used in the proof of Robertson-Seymour minor theorem (Theorem 2.4). We refer to [12] as a source for the contents of this section, where an interested reader can find more details. We omit all proofs due to their complexity.

Proposition 3.3 implies that every class of graphs of treewidth bounded by a constant is minor-closed. The following generalization of Kruskal Theorem was one of the first basic steps in the proof of Robertson-Seymour theorem.

Theorem 3.23. *For every k, the class of graphs of treewidth bounded by k is well quasi-ordered by the minor relation.*

We showed in Section 2 the case $k = 1$. The case of an arbitrary k is solved by k iterations of arguments used in the proof of Kruskal theorem.

Conversely, we can characterize those classes of graphs that have bounded treewidth and exclude a certain minor.

Theorem 3.24. *For every graph H the following equivalence holds: H is planar if and only if the class of graphs which do not contain H as a minor has bounded treewidth.*

Since every planar graph is a minor of a grid of sufficient size, it is sufficient to prove:

Theorem 3.25. *For every integer n exists k_n such that every graph of treewidth at least k_n contains the $n \times n$ grid as a minor.*

Based on these results we are ready to briefly describe the idea of the proof of Theorem 2.4. Consider a sequence of finite graphs $G_0, G_1, \ldots$. We find a good pair $G_i \preceq G_j$, $i < j$ as follows. If G_0 is planar and if it is not a minor of the forthcoming graphs then graphs in the rest of the sequence have bounded treewidth. Since they are well quasi-ordered a good pair exists.

If G_0 is not planar then we may assume that none of $G_1, G_2, \ldots$ contains $K_{|V(G_0)|}$ as a minor. Such graphs can be characterized by their tree decompositions and by their embedding into surfaces of bounded genus. Robertson and Seymour proved that these graphs are also well quasi-ordered by the minor relation.

Finally, we show weaker version of Theorem 3.25, restricted to planar graphs.

Theorem 3.26 (Alon, Seymour and Thomas)**.** *Every* planar *graph of treewidth at least $20k - 12$ with contains the $k \times k$ grid as a minor.*

The proof is adapted from [32].

Proof. By Corollary 3.18 G contains a strongly $(5k - 3)$-linked set W^*.

We now fix a planar drawing of G and find a simple (i.e. non-selfintersecting) closed curve φ subject to the following conditions:

- φ intersects the drawing of G only in vertices,
- φ contains at most $4k$ vertices of G,
- the internal face (without the boundary) of φ contains more than $\frac{2}{3}|W^*|$ vertices of W^*,

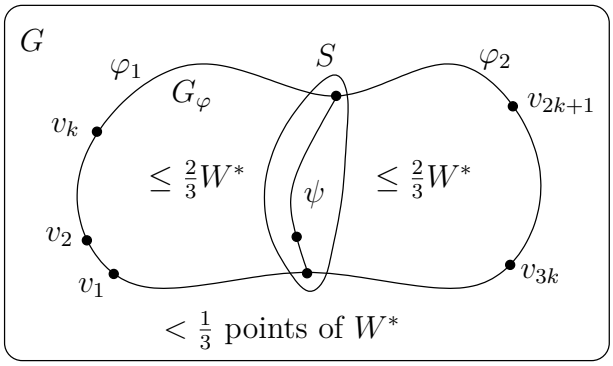


Figure 12: Vertices on $\varphi \cup \psi$ form a separator showing that W^* is not strongly $(5k-3)$-linked.

- subject to the above conditions the internal face of φ contains as few vertices of G as possible.

Such curve φ exists, since a circle around G fulfills the first three necessary conditions. Observe that φ contains exactly $4k$ vertices of G: If two adjacent vertices are consecutive on φ, we draw φ such that the edge is in the external face of φ. Then φ can be adjusted to accommodate further vertices from the adjoining faces of G.

Let us denote the $4k$ vertices along φ by $v_1, \ldots, v_{4k}$. We show that there are k vertex disjoint paths $R_1, \ldots, R_k$ between pairs v_1 and v_{3k}; v_2 and v_{3k-1}; $\ldots$; v_k and v_{2k+1}.

Assume the opposite. By a planar version of Menger's theorem there is a cutset S of size at most $k-1$ for the subgraph G_φ of G induced by the vertices of the internal face of φ. Note that S contains at least two vertices lying on φ. Let φ_1 and φ_2 be the two fractions of the curve φ.

We draw a curve ψ through the cutset S and obtain two closed curves $\varphi_1 \cup \psi$ and $\varphi_2 \cup \psi$. As both have at most $4k$ vertices, their internal faces contain at most $\frac{2}{3}|W^*|$ vertices of W^* — otherwise it would contradict the choice of φ. But by Observation 3.16 the set of vertices lying on $\varphi \cup \psi$ is a good separator contradicting the fact that W^* is strongly $(5k-3)$-linked, see Fig. 12.

Analogously we find k vertex disjoint paths $C_1, \ldots, C_k$ between pairs v_{k+1} and v_{4k}; $\ldots$; v_{2k} and v_{3k+1}.

We form the $k \times k$ grid from the union of paths $R_1, \ldots, R_k$ (considered as rows) and $C_1, \ldots, C_k$ (considered as columns). Adjust the drawing such that rows are straight horizontal lines ordered from top to bottom and columns are ordered left to right.

Whenever possible, we purge edges with a vertex of degree one, and contract one of two consecutive edges of the same row or column, resp., if the middle vertex is of degree two.

We sweep the drawing left to right and adjust these paths so that each column becomes monotone, i.e., each edge traverses to a highly numbered row. By induction, assume that $C_1, \ldots, C_{i-1}$ are already adjusted. From C_i first select all leftmost edges that are monotone.

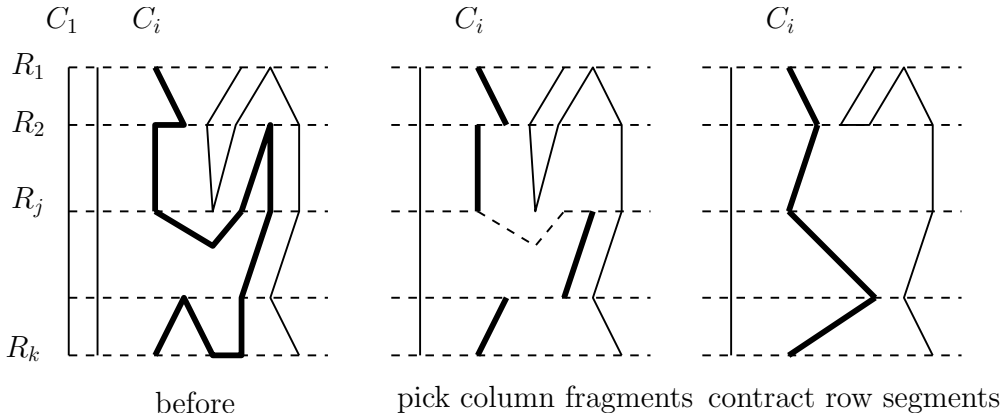


Figure 13: Construction of the $k \times k$ grid.

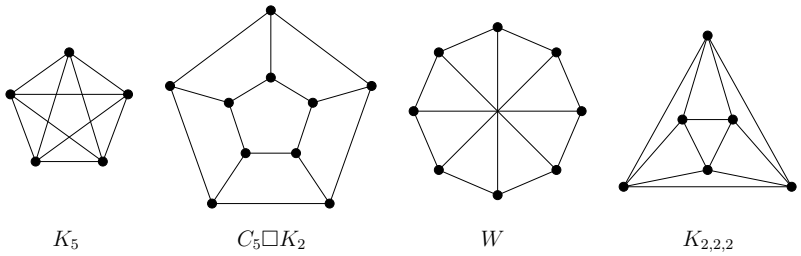


Figure 14: Forbidden minors for graphs of treewidth at most three.

If this set forms a path, we are done. Otherwise we contract the row segments between the chosen fragments of the path C_i.

It may happen that some row segment contains vertices of sequel columns. In such a case the row should be adjusted to bypass these vertices (see the situation depicted in Fig. 13).

The resulting graph is a $k \times k$ grid minor of the original graph G. □

Finally, we show that some classes of graphs of bounded treewidth have known a good characterization in terms of forbidden minors [5].

Theorem 3.27. *The treewidth of a graph is*

- *at most one if and only if it does not contain $K_3 = C_3$ as a minor.*
- *at most two iff it contains no K_4 minor.*
- *at most three iff no graph depicted in Fig. 14 is its minor.*

Observe that the first case describes forests. Subgraphs of series-parallel graphs correspond to the second case by Theorem 3.6.

We conclude this section by one of the core conjectures of structural graph theory:

Conjecture 3.28 (Hadwiger, 1943)**.** *The chromatic number $\chi(G)$ of an arbitrary graph G is at most the size of the largest complete graph which is a minor of G.*

Define the Hadwiger number $h(G)$ as the size of the largest complete graph which is a minor of G.

When $h(G) \leq 2$ (i.e. K_3 is the forbidden minor) then the graph is a forest. Forests are bipartite, hence 2-colorable.

Every series-parallel graph has a vertex of degree at most two. Hence graphs with no K_4 minor are 2-degenerate, and consequently 3-colorable, which yields the case for $h(G) = 3$.

Every planar graph has forbidden K_5, hence Hadwiger's conjecture for $h(G) = 4$ implies four color theorem. Wagner indeed showed in 1937 their equivalence. The idea is that every graph without K_5 as a minor can be decomposed into pieces that are planar or thar are isomorphic to the Wagner graph W depicted in Fig. 14.

The conjecture was proved also for graphs with $h(G) = 5$. Robertson, Seymour and Thomas [36] showed that this case is equivalent to the four color theorem as well.

A *k-decomposable* graph allows an ordering of its vertices, such that each vertex is adjacent to at most k its successors. Obviously, every partial k-tree is k-decomposable.

Exercise 37: Show that k-decomposable graphs are not necessarily subgraphs of k-trees.

Degeneracy of a graph G is defined as the maximum of the minimum degree $\delta(H)$ taken over all induced subgraphs $H \subseteq G$. In other words, k-decomposable graphs are graphs of degeneracy at most k.

Exercise 38: Show that the degeneracy of a graph is a lower bound on its treewidth.

4 Further graph decompositions

4.1 Nice tree decomposition

A better structured tree-decompositions might be often helpful, especially in the algorithm design.

Definition. The *nice tree decomposition* of G ([17]) is a tree decomposition where T is a rooted binary tree and for each inner node X_i it holds that:

- either X_i has two children X_j and $X_{j'}$, where $X_j = X_{j'} = X_i$,
- or X_i has one child X_j, where sets X_i and X_j differ by exactly one vertex.

Sometimes it is required that all leaves in T contain only one vertex of G. We adopt the following terminology: Nodes with two children are *join* nodes. A node that has an additional vertex compared to is only child is an *introduce* node. If in such situation some vertex is missing, then it is is a *forget* node.

Observation 4.1. *Any tree decomposition can be transformed in polynomial time into a nice tree decomposition of the same width.*

Proof. Pick a root arbitrarily and then perform further local replacements. Each node X_i of degree at least four will be substituted by at $\deg_T(X_i) - 2$ union nodes, each edge will be substituted by at most k forget and at most k indroduce nodes, where k is the width of T.

If leaves should be of size one, we replace current leaves by paths of introduce nodes of length at most k. □

Corollary 4.2. *Each graph G has a nice tree decomposition on $O(|V_G|\mathrm{tw}(G))$ nodes with all leaves of size one.*

Proof. Apply Observation 4.1 on an optimal tree decomposition on at most n nodes. □

We show that the constuction of Observation 4.1 is asymptotically tight:

Proposition 4.3. *For every k there exist a graph on $3k$ vertices whose any optimal nice tree decomposition with leaves of size one has $\Omega(k^2)$ nodes.*

Proof. We first build an auxiliary tree consisting of a matching on k edges (u_i, v_i) and a vertex r connected to all vertices v_i. To form G we further add k universal vertices $w_1, \ldots, w_{k-1}$. Straightforwardly $tw(G) = k$.

Take a nice optimal tree decomposition T of G. Each set $\{u_i, v_i, w_1, \ldots, w_{k-1}\}$ induces a clique in G and hence forms a node of X_i of T. Observe that only one component of $T \setminus X_i$ contains nodes with r, while nodes of the other components are subsets of X_i.

The minimum subtree of T containing nodes $X_1, \ldots, X_k$ is still valid (non-nice) tree decomposition of G, where each X_i for $i = 1, \ldots, k$ is a leaf. Only at most one such leaf, say X_1, can be predecessor of the other nodes $X_2, \ldots, X_k$ in T. Therefore, the nodes $X_2, \ldots, X_{k-1}$ are in no predecessor-successor relation.

Finally, since all leaves of T should be of size one, each subtree of T rooted in X_i for $i = 2, \ldots, k-1$ has at least $k+1$ introduce or leaf nodes. Hence, the required decomposition contains at least $(k-2)(k+1)$ nodes. □

Finally we prove that for ordinary nice tree decompositions (with leaves of any size), the number of nodes can be reduced.

Proposition 4.4 ([17])**.** *Any graph G allows an optimal nice tree decomposition with at most $4|V_G|$ nodes.*

Proof. Without loss of generality we may assume that G is a k-tree, and hence by Proposition 3.8 it contains a vertex u whose neigborhood $N(u)$ induce a clique of size k.

Assume by induction that T' is a suitable decomposition of $G - u$. Here, let X_i be a node of T' containing $N(u)$ as a subset.

If X_i is a join node, then we use one of its children instead of it. Hence we may assume that X_i has at most one child.

If X_i is not a leaf, then we fist subdivide the edge from its parent to X_i by a new join node X_j. The new join node X_j will also have a chid $X_{j'}$ which is a leaf. Now we use this leaf instead of X_i.

Finally, if X_i is a leaf, then we promote it into a introduce node by adding one forget-type child $X_j = N(u)$ and grandchild $X_{j'} = N(u) \cup \{u\}$. □

4.2 Treewidth versus pathwidth

A path decomposition of a graph is a tree decomposition where the underlying tree forms a path. The notion of the pathwidth is defined analogously. Clearly $\mathrm{tw}(G) \leq \mathrm{pw}(G)$ for any graph G, because a path decomposition can be viewed as a tree decomposition. On the other hand, graphs with bounded treewidth may attain arbitrarily high pathwidth.

Proposition 4.5. *The pathwidth of the rooted ternary tree T_k with $k+1$ levels is at least k.*

Proof. We proceed by induction. Let P be a path decomposition of $T = T_k$ and let T^1, T^2 and T^3 be the three trees isomorphic to T_{k-1} rooted at the three children of the root of T. Without loss of generality we may assume that the leftmost node of P contains some vertices of T^1. and that the rightmost node contains some vertices of T^1 or of T^3. Then all nodes of P that contain some vertices of T^2 include also some vertex of $T \setminus T^2$, hence $\mathrm{pw}(T) \geq \mathrm{pw}(T^2) + 1$. □

Theorem 4.6. *For every graph G on n vertices it holds that* $\mathrm{pw}(G) = O(\mathrm{tw}(G)\ln(n))$.

Proof. Consider a rooted tree decomposition T of G of width $\mathrm{tw}(G)$ with at most n nodes. Such a decomposition can be derived from an embedding of G into a k-tree.

We first define an auxiliary labeling l of the tree T as follows: Put 1 as the label for every leaf of T. If an internal node X_i has only one child X_j, we set $l(X_i) := l(X_j)$. If X_i has more children $X_j, X_{j'}, X_{j''}, \ldots$ ordered nonincreasingly by their labels, then we select $l(X_i) := \max\{l(X_j), l(X_{j'}) + 1\}$. Observe that the root has label at most $\log_2(n) + 1$, since $l(X_i)$ is greater than $l(X_j)$ if and only if $l(X_j) = l(X_{j'})$. By induction we can derive that a tree whose root is labeled by i contains at least $2^i - 1$ nodes.

We now define a path decomposition P according to the labeling of T recursively. For any subtree T_i rooted in a node X_i we define a path decomposition P_i of G_i, i.e. of the subgraph of G induced by the vertices of $\bigcup_{X \in T_i}$.

A leaf X_i of T is viewed as a single-node path decomposition of G_i. If a node X_i has only one children X_j, then we attach X_i to the path decomposition P_j. If X_i has more children then we construct the path decomposition P_i as the concatenation of $P_j \circ P' \circ P'' \circ \cdots \circ X_i$, where $P^{(s)}$ is the path decomposition of $G_{j^{(s)}}$, where each node is extended by the set X_i.

These decompositions have the property that X_i is always the last node of P_i, hence are valid path decompositions.

Observe that for any node Y of P_i the number of nodes from T, that were used to build Y, is bounded by $l(X_i)$. Hence, the width of any path decomposition P_i of G is bounded by $l(X_i)\mathrm{tw}(G)$. This yields that $pw(G) \leq (\log_2(n) + 1)(\mathrm{tw}(G) + 1) - 1$. □

Note that this upper bound is asymptotically tight — from the previous proposition we have $\mathrm{pw}(T_k) \geq k = \lceil \log_3 |V_{T_k}| \rceil - 1$.

Observe that graphs of pathwidth at most k are exactly those graphs where $k+1$ police helicopters can capture an invisible robber. In other words police helicopters move without knowing the robber's position and the game ends when the robber cannot move.

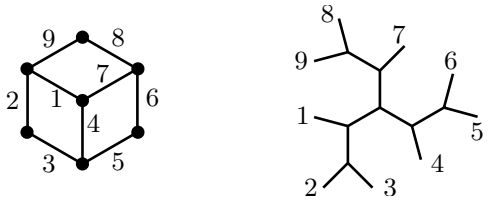


Figure 15: An example of a graph and its branch decomposition of width 3.

4.3 Treewidth versus branchwidth

The notion of branchwidth in fact preceded the notion of treewidth. It is defined as follows:

Definition. A *branch decomposition* of a graph G with at least two edges is a tree B where each internal node has degree three and leaves of B are in one-to-one correspondence with edges of G. Each edge $f \in E_B$ splits E_G into the two sets E_f and $E'_f = E_G \setminus E_f$ where the edges of E_f are indicated by the leaves of one of the two components of $B \setminus f$.

The width of an edge f is the number of vertices that appear both in E_f and in E'_f as an endvertex of some edge. Formally, $w(f) = |V_f| = |\{(\bigcup E_f) \cap (\bigcup E'_f)\}|$. The width of a branch decomposition B is the maximum width among all its edges and the *branchwidth* of G, denoted by $\mathrm{bw}(G)$, is the minimum width out of all possible branch decompositions of G.

The branchwidth of a graph with at most one edge is considered to be 0.

Exercise 39: Characterize graphs of branchwidth one.

Exercise 40: For each tree, construct a branch decomposition of width at most two.

Exercise 41: Show that $\mathrm{bw}(H) \leq \mathrm{bw}(G)$ whenever H is a minor of G.

Exercise 42: Show that the branchwidth of the complete graph K_n on at least 3 vertices is at least $\lceil \frac{2}{3}n \rceil$.

Theorem 4.7. *For any graph G of branchwidth at least two it holds that* $\mathrm{bw}(G) \leq \mathrm{tw}(G) + 1 \leq \lfloor \frac{3}{2}\mathrm{bw}(G) \rfloor$.

Proof. Let T be a tree decomposition of G of width $\mathrm{tw}(G)$. For each edge $e \in E_G$ select a node X and join X with an exposed leaf representing e.

We now take the smallest subtree containing all the exposed leaves and contract all vertices of degree 2 and possibly replace internal vertices of higher degree by ternary trees. We claim that B has width at most $\mathrm{tw}(G) + 1$.

If $f \in E_B$ is incident with a node X of the original tree decomposition then $V_f \subseteq X$, since other vertices are separated by X (from E_f or E'_f), and cannot contribute to V_f. This establish the first inequality.

Let B be an arbitrary branch decomposition of G of width $\mathrm{bw}(G)$. We construct a tree decomposition $T = B$ of G as follows: If a leaf X_e corresponds in B to the edge e of G we let $X_e = e = \{u, v\}$. For each pair of edges $e, e' \in E_G$ with a common vertex u we

insert u into all internal nodes on the unique path between X_e and $X_{e'}$. This is a tree decomposition of G of width at most $\lfloor \frac{3}{2}\mathrm{bw}(G) \rfloor$, because each vertex is inserted twice in each internal node. □

4.4 Treewidth versus NLC-width

The major disadvantage of all already discussed decompositions is that they cannot capture well some classes of graphs, especially when they contain large cliques, e.g. complete graphs or cographs.

This is possible in the notions of cliquewidth [10], node label controlled width (NLC-width) [41] or rankwidth [28]. Similarly as in the previous cases a graph is described in a tree-like manner.

Definition. Let $I = \{1, \ldots, k\}$ be a set of labels. A *expression tree* of a graph G is a binary tree E of the following form:

- The nodes of E are of the three types i, $\oplus$ and ρ:
- Introduce nodes $i(v)$ are leaves of E, corresponding to a graph, with only one vertex v of label $i \in I$.
- A join node $\oplus_S$, where S is a binary relation $S \subseteq I \times I$, stands for a disjoint union of the two graphs associated with its two children, where for each pair $(i, j) \in S$ all edges are added between vertices labeled by i of the left child and vertices labeled by j of the right child.
- A relabel node ρ_r, where r is a mapping $r : I \to I$ is associated with the graph of its only child, where in addition each vertex label i is replaced by the label $r(i)$.
- The graph G is isomorphic to the graph associated with the root of E with all vertex labels removed.

The size of the minimum set of labels necessary to construct an expression tree for a graph G is denoted by $\mathrm{nlcw}(G)$ and is called the *NLC-width* of G.

Exercise 43: Determine the NLC-width of trees and of complete graphs.

Exercise 44: Show that graph of cliquewidth one are so called complement-reducible graphs (shortly cographs). i.e. graphs G where G or $\overline{G}$ are disconnected and where the components are cographs.

Cographs are known to be exactly those graphs with no induced path of length four.

Exercise 45: Show that graphs of bounded degree and bounded NLC-width have bounded treewidth.

We show that graphs of bounded treewidth have bounded NLC-width.

Theorem 4.8. *For any graph G it holds that* $\mathrm{nlcw}(G) \leq 2^{\mathrm{tw}(G)+1} + \mathrm{tw}(G) + 1$.

Proof. Let a graph G have a nice tree decomposition T of width k with root node X_r. We construct by induction an expression tree E that can be split into three parts:

- a subtree E_r for G_r, the subgraph of G induced by X_r,
- a subtree E' corresponding to $G' = G \setminus G_r$,
- a union node $\oplus_S$ above E_r and E', where S describes the edges between X_r and $V_G \setminus X_r$.

The labels on E are chosen as follows: vertices on G_r have $k+1$ distinct labels, different from the labels used on G'. Vertices in G' are labeled such that vertices of the same label have the same neighbors in X_r (could be empty as well).

In total we need $k+1$ labels for G_r and at most 2^{k+1} labels for G'.

We now mimic the construction of E as was shown above for the case of trees. If X_r is

- the leaf node corresponding to a vertex v, we take $E = \{1(v)\}$,
- the introduce node, we modify only E_r (the child has at most k vertices, so we may use a new label),
- the forget node for a vertex v — we remove v from E_r, and extend E' such that v is (see Fig. 16):
  - introduced with the same label as it was in G_r,
  - joined to E' by $\oplus_S$ node (as labels on G_r and G' are distinct, we may assume that S is the same on all nodes inside E_r and that it describes also all edges between G_r and G'),
  - relabeled to j, i.e. the label of vertices in G that have the same neighbors in X_r as v.

  We finally merge some labels in $G' \cup v$ for the vertices which have the same neighborhood in $X_r \setminus v$.
- the union node with respect to subtrees T' and T''. Observe that $V_{G'}$ must be disjoint from $V_{G''}$. Without loss of generality we may assume that vertices of G' and G'' have the same label iff they have the same set of neighbors in G_r, in particular, the relation S is the same the root $\oplus_S$ of both trees T' and T''. Then we place $E' \oplus E''$ below the union node with E_r (see Fig. 17).

In total we use $2^{k+1} + k + 1$ labels and the claim follows. □

Expression trees for the closely related graph parameter called *cliquewidth* have the following differences:

- A relabel node $\rho_{i \to j}$ is associated with the graph of its only child, where in addition each vertex label i is replaced by the label j.

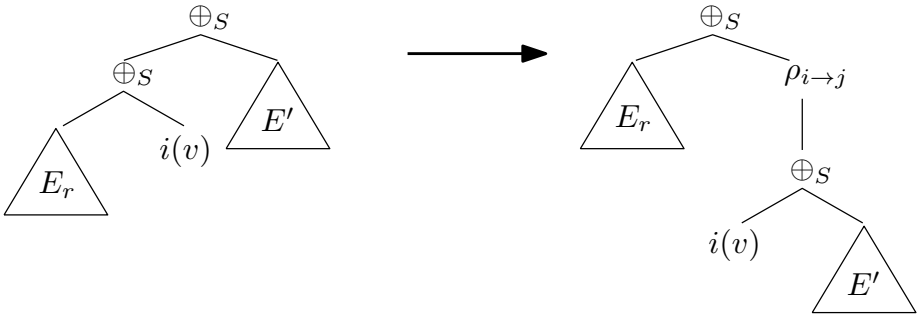


Figure 16: Transformation of a tree decomposition into an expression tree — forget node.

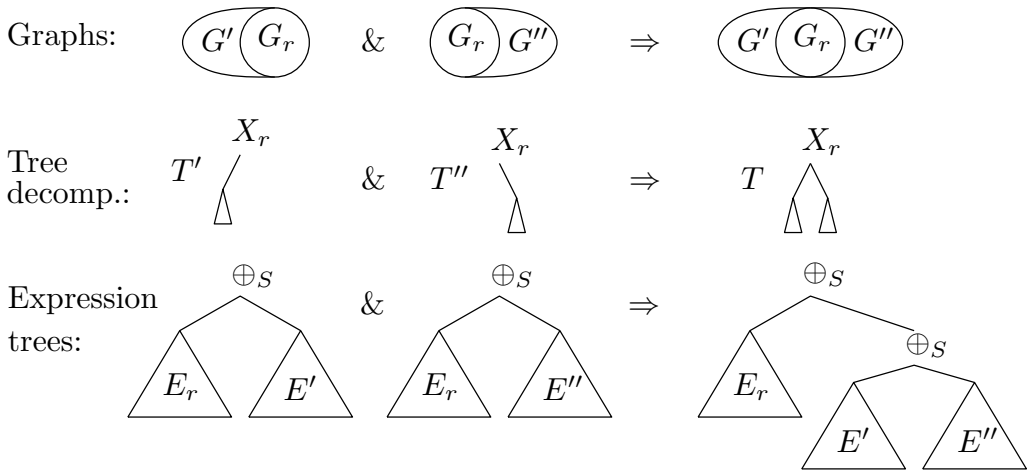


Figure 17: Transformation of a tree decomposition into an expression tree — union node.

- A node $\oplus$ stands only for the disjoint union of the two graphs associated with its two children (no edges are added).
- A new type of node $\eta_{i,j}$ is introduced, where $i, j \in I, i \neq j$ and the associated graph is obtained by joining nodes of label i to the nodes of label j in the graph corresponding to the only child of the node.

It is possible to construct graphs whose cliquewidth is exponential in treewidth. In particular there exists a k-tree with cliquewidth at least $2^{\lfloor k/2 \rfloor - 1}$ [9].

Exercise 46: Determine the relation between $\mathrm{cw}(G)$ and $\mathrm{nlcw}(G)$.

4.5 NLC-width versus rankwidth

Rankwidth is another graph parameter introduced by Oum and Seymour [28]. The definition of rankwidth is similar to the definition of branchwidth:

Definition. A rank decomposition of a graph G is a ternary tree R whose leaves are in bijection with vertices of G. Each edge $f \in E_R$ corresponds to a partition of vertices of G into two sets V_f and $\overline{V_f}$. The width of f is the rank of the adjacency matrix A_f between V_f and $\overline{V_f}$ over the field $GF(2)$.

The width of the decomposition R is the maximum width of its edges, and the rankwidth $\mathrm{rw}(G)$ of G is the minimum possible width among its rank decompositions.

Exercise 47: Determine the rankwidth of a complete graph.

It is known that $\mathrm{rw}(G) = 1$ if and only if G is distance-heriditary [26], i.e. graphs, where distances in each induced connected subgraph are the same as in the original graph.

Proposition 4.9 ([28])**.** *For any graph G it holds that*

$$\mathrm{rw}(G) \leq \mathrm{nlcw}(G) \leq 2^{\mathrm{rw}(G)}.$$

Proof. In order to prove the first inequality we transfom an expression tree E into a rank decomposition R in a straightforward way — we contract all relabel nodes.

The leaves of R are in one-to-one correspondence to vertices of G as they were introduced in E. It suffices to observe that for any edge $f \in E_R$ either the set V_f or $\overline{V_f}$ correspond to the vertex set of a labeled graph given by a subtree of E. Assume without loss of generality that V_f is such set.

Vertices with the same labels in V_f will be treated uniformly during join operations, hence they have the same neighbors in $\overline{V_f}$. The rank of the adjacency matrix is bounded by the number of its distinct rows. As vertices of V_f of the same label provide identical rows, the first inequality follows.

For the other bound we convert a rank decomposition R into an expression tree E. We subdivide any edge of R by an extra new vertex and promote it to the root of the tree. Now leaves are in bijection with vertices of G and internal nodes can be viwed as union

nodes. It remains to implement labels and join nodes to introduce edges into the graph. Let k be the width of the rank decomposition R.

For an internal node $X_i \in V_R$, let G_i be the subgraph of G induced by the vertices corresponding to descendants of X_i. Assume that $V_{G_i} = V_f$ for the edge f leaving from X_i to its parent (the root node is treated similarly).

We choose a labeling of $V(G_i)$ such that vertices u and v share the same label if and only if they have the same neighbors in $V_G \setminus V_{G_i}$. Observe that the same labels should be assigned to vertices corresponding to the same rows in the associated adjacency matrix. As the matrix may have at most 2^k distinct rows, we get that 2^k distinct labels suffices for the desired labeling.

Let X_i be the parent of X_j and X_k and assume by induction that E_j and E_k are expression trees for the labeled graphs G_j and G_k. We construct an expression tree E_i of G_i as follows:

- join the two trees by $\oplus_S$, where S describes all edges between vertices of V_{G_j} and V_{G_k},
- add a relabel node to get the same label for verices that have the same neighbors outside G_i.

To argue feasibility of the second step note that by the choice of the labeling we know that vertices of the same label in G_j could not be distinguished by the neighbors outside G_j. Observe that their neighborhood outside G_i is even more restricted, hence these vertices should have identical labels also in G_i (and vice-versa for vertices of G_k).

Straightforwardly, we have used at most $2^{k+1} - 1$ distinct labels in E_i. □

Definition. Let v be a vertex of G, then by a *local complementation* of v in G we get a graph $G * v$ that differs from G only by edges on $N(v)$. Namely, the induced subgraph $G|_{N(v)}$ is replaced by its complement $\overline{G|_{N(v)}}$.

We say that H is a *vertex minor* of G if it can be obtained by vertex deletions and local complementations.

Proposition 4.10 (Oum [26])**.** *The local complementation does not alter the width of any edge in a rank decomposition.*

Proof. Let v be any vertex of G and let f be any edge of a rank decomposition of G. Without loss of generality we may assume that $v \in V_f$.

We alter the matrix A_f by adding the row corresponding to v to all rows corresponding to vertices on $N(v) \cap V_f$. As the result is the adjacency matrix of $G * v$ with respect to the same partition V_f and $\overline{V_f}$, the claim follows straightforwardly. □

Corollary 4.11. *If H is a vertex-minor of G then* $\mathrm{rw}(H) \leq \mathrm{rw}(G)$.

Oum proved that:

Theorem 4.12 ([27])**.** *The class of graphs of bounded rankwidth is well-quasi ordered by the vertex-minor relation.*

The proof is complicated and omitted here.

Compare the last theorem with Theorem 3.23 of Robertson and Seymour, which shows that the minor relation also provide a well-quasi ordering on more restricted graph classes (bounded treewidth yields bounded rankwidth by Theorem 4.8 and Proposition 4.9).

5 Algorithms for bounded treewidth graphs

5.1 Computing treewidth

The problem to determine whether a given graph G has treewidth at most k is NP-complete [4].

On the other hand for fixed k the problem $\mathrm{tw}(G) \leq k$ could be decided in linear time [6].

Here we prove Theorem 3.17 and present a simple quadratic algorithm that either answers that the $\mathrm{tw}(G) > k$ or outputs a tree-decomposition of width at most $4k+3$ [30]. Then the exact treewidth of G could be computed by the technique described in Section 5.2.

The basic idea is that we try to find recursively a good separator S of size at most k for a set W whose size is between $2k+3$ and $3k+3$. If no such good separator exists then, by Lemma 3.15, the treewidth of G is greater than k. If we succeed in all recursive steps, we construct a tree decomposition of width at most $4k+3$ as depicted in Fig. 18.

The code of the algorithm follows. To obtain the decomposition of the entire graph G we choose for W any set of size $3k+3$.

```
TREE DECOMPOSITION (G, W)
Input: A graph G, tw(G) ≤ k, a set W ⊆ V_G, |W| ≤ 3k + 3.
Output: A rooted tree decomposition T of G of width at most 4k + 3,
        where W ⊆ root(T).
begin
  if |V_G| ≤ 4k + 4 then let T has the only node V_G; exit
    else
      if |W| ≤ 2k + 2 then augment W arbitrarily to the size 2k + 3 endif
      find a good separator S for W in G of size at most k + 1
      for I = A, B where A and B are the two sets separated by S
          G_I := subgraph of G induced by I ∪ S
          W_I := S ∪ (W ∩ I)
          T_I :=TREE DECOMPOSITION (G_I, W_I); endfor
      T := tree rooted at {W ∪ S} with subtrees T_A and T_B endif
end.
```

The correctness of the algorithm follows from the following arguments.

The size of the set W_I is bounded, since

$$|W_I| \leq |S| + \frac{2}{3} \cdot |W \setminus S| \leq k + 1 + \frac{2}{3} \cdot (3k+3) = 3k+3.$$

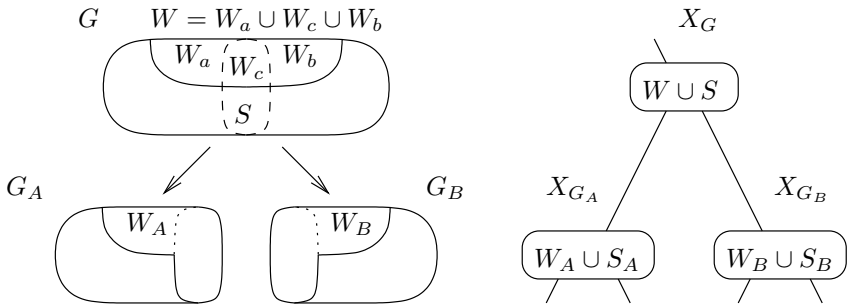


Figure 18: Recursive finding good separators and building tree decomposition.

The constructed tree decomposition T is of width at most $|W| + |S| - 1 \leq 4k + 3$. We have obtained a proper tree decomposition, since every edge of G will be represented either in the node $W \cup S$ or in one of recursive trees for G_I. Moreover all nodes containing a fixed vertex form a connected subtree, since the only vertices that appear simultaneously in the nodes of T_A and T_B are the elements of S and hence appear in particular in the tree nodes X_G, X_{G_A} and X_{G_B} (consult Fig. 18).

Now the only missing step is to find good separators quickly:

Lemma 5.1. *For any graph G with n vertices and at most kn edges, and for any $W \subseteq V_G$ of size at most $3k + 3$, it is possible to find a good separator S for W of size at most $k + 1$ or answer that no such good separator exists in time $O(27^k n)$.*

Proof. From the definition of a good separator S for W follows that W should be partitioned into three disjoint sets W_a, W_b and W_c, where $|W_a|, |W_b| \leq \frac{2}{3}|W|$, and where W_c is a subset of a separator of size at most $k + 1$ separating A and B, where $W_a \subseteq A$ and $W_b \subseteq B$ (consult Fig. 18).

Observe first that W_c can be extended to such desired separator if and only if in the graph $G \setminus W_c$ there are at most $k + 1 - |W_c|$ vertex disjoint paths between sets W_a and W_b.

Hence, we try all possible partitions of W into three parts W_a, W_b and W_c with appropriate sizes. There are at most 3^{3k+3} possibilities.

For each partition we solve the vertex-disjoint path problem. If all tests fail then no good separator for W exists..

Note that the existence of the disjoint paths could be decided by the maximum flow technique. It needs at most $k + 1$ iterations of finding an augmenting path in a graph with $O(kn)$ edges. This step requires $O(k^2 n)$ time. □

The time complexity of the Tree decomposition algorithm composes of at most n iterations of the Separator search algorithm, because the final tree-decomposition has at most n nodes. In total we get $O(27^k n^2)$ required operations.

5.2 More applications and nice decompositions

In this section we show how some computationally hard problems allow a tractable solution when instances are restricted to graphs of bounded treewidth.

The general scheme is based on a "brute force" enumeration of all feasible solutions of the given problem when restricted onto graph G_i, i.e. the subgraph of G induced by the node X_i. We assume that the tree T is rooted, and for each node X_i we build a table $\texttt{Tab}_i$ of constant or polynomial size, whose entries describe necessary properties of partial solutions on the graph G_i, and sometimes also on the graph G'_i, that is the subgraph of G induced by the union of X_i and all its descendants. The data entries for X_i should be computed in polynomial time from the entries in $\texttt{Tab}_j$, corresponding the children X_j of X_i.

The situation when the table $\texttt{Tab}_m$ is nonempty for the root node X_m usually indicates that a proper solution of the entire graph G exists.

We illustrate this approach on the problem of computing the size of the maximal independent set $\alpha(G)$.

Each entry in $\texttt{Tab}_i$ is the pair (S, a), where S is an independent set in G_i (possibly empty) and a is the size of the maximum independent set of G'_i with S as its intersection with X_i.

In the following algorithm we call an independent set S_j of G_j feasible for S_i if $X_j \cap S_i = X_i \cap S_j$.

The algorithm works as follows:

MAXIMUM INDEPENDENT SET
Input: *A tree decomposition of a graph G of fixed width k.*
Output: *Size of the maximal independent set $\alpha(G)$.*
begin
  mark all nodes as unfinished
  while exists an unfinished node X_i with all children finished **do**
    begin
      for all independent sets S_i in G_i **do**
        if for all children X_j of X_i there exists
          a set S_j feasible for S_i, s.t. $(S_j, a_j) \in \texttt{Tab}_j$
          then
            begin
              for each X_j select a S_j feasible for S_i
                s.t. $(S_j, a_j) \in \texttt{Tab}_j$ and a_j is maximized
              $a_i := |S_i| + \sum_j (a_j - |S_j \cap S_i|)$
              store (S_i, a_i) into $\texttt{Tab}_i$
            end;
      mark X_i as finished
    end;
  $\alpha(G) = \max\{a_m : (S_m, a_m) \in \texttt{Tab}_m\}$.
end.

Observe that for any X_i the computation of the table $\texttt{Tab}_i$ requires at most $O(2^{2|X_i|})$ operations. By assuming that the treewidth of G is bounded by a constant, the algorithm

runs in time $O(|T|)$ which is function linear in n.

Exercise 48: Modify the algorithm such that it outputs also an independent set of size $\alpha(G)$.

We use nice decompositions to solve efficiently the graph coloring problem, e.g. a decision problem, testing whether $\chi(G) \leq c$. Observe that $\chi(G) \leq \mathrm{tw}(G) + 1$, so we can assume $c \leq \mathrm{tw}(G)$.

Now the table $\mathtt{Tab}_i$ stores all colorings (viewed as vertex partitions) of G_i with at most c colors which allow an extension to G'_i.

```
c-Colorability
Input: A tree decomposition of a graph G of width k, an integer c.
Output: Decision whether χ(G) ≤ c.
begin
  if c ≥ k then return "YES" and quit
  mark all nodes as unfinished
  for all leaves X_i put all colorings φ of G_i with at most
    c color classes into Tab_i and mark X_i as finished
  while exists an unfinished node X_i with all children finished do
    begin
      case X_i of
        — introduce node with child X_j and v = X_i \ X_j
          then store in Tab_i all extensions φ_i of φ_j ∈ Tab_j onto G_i
            where no neighbor of v is in the same color class as v
            and φ_i has at most c color classes
        — forget node with child X_j,
          then store in Tab_i all φ_j ∈ Tab_j restricted to X_i
            and remove duplicate entries
        — join node with children X_j, X_j'
          then Tab_i = Tab_j ∩ Tab_j'
        esac
      mark X_i as finished
    end;
  return "YES" if Tab_root ≠ ∅ and "NO" otherwise.
end.
```

Exercise 49: Modify the algorithm such that it finds $\chi(G)$ and outputs also an optimal coloring of G.

Exercise 50: Design an algorithm for testing whether a graph of fixed treewidth c is Hamiltonian, provided that the tree-decomposition of width at most c is a part of the input.

5.3 Properties expressible in MSO

The fact that several graph properties can be tested in linear time on graphs of bounded treewidth was generalized by Courcelle to all graph properties expressible in monadic second order logic (MSO). We recall that in the first order logic a formula allows quantifiers over single elements, while in the second order logic a quantification over predicates is allowed. The monadic second order logic restricts predicates only to unary predicates, i.e. to sets of elements.

A graph G may be modelled in a predicate logic in several ways. E.g. the universe may consists of vertices of G and the edges are given as the binary adjacency relation. This graph model in MSO is often denoted as MSO_1. Another way is to use the edges and vertices as the universe and describe the graph by the binary incidence relation. This model is often denoted as MSO_2. We show that since the quantifiers may range also over sets of edges, more graph properties may be expressed in this logic.

In summary MSO_1 formuli for testing graph properties may contain:

- the usual logical conenctives $\neg, \wedge, \vee, \Leftrightarrow, \Rightarrow \ldots$.
- variables for vertices and vertex sets
- predicates for the membership $\in$, equality $=$, and for the adjacency relation
- constants: particular vertices or subsets of V_G, e.g. $\emptyset$

The MSO_2 allows variables and constants for edge sets, and also quantification over edge sets.

As an example we show a MSO_2 formula for testing Hamiltonian graphs:

$$
\begin{aligned}
&\exists E \subseteq E_G : \\
&[\forall u \in V_G \, \exists v, w \in V_G : (v \neq w) \wedge ((u,v) \in E) \wedge ((u,w) \in E) \wedge \\
&\qquad \wedge [\forall x \in V_G : (u,x) \in E \Rightarrow ((x = v) \vee (x = w))]] \\
&\qquad\qquad \ldots E \textit{ is a 2-factor} \\
&\wedge [\forall V \subseteq V_G : ((V \neq V_G) \wedge (\exists u \in V_G : u \in V)) \Rightarrow \\
&\qquad \Rightarrow (\exists v, w \in V_G : (v,w) \in E \wedge (v \in V) \wedge (w \notin V))] \\
&\qquad\qquad \ldots \textit{and it is connected}
\end{aligned}
$$

We note that the graph property of being Hamiltonian is not expressible in MSO_1. This could be argued as follows. Words of length n over alphabet $\{a, b\}$ can be modeled as linearly ordered sets of size n with one unary predicate P_a indication on which positions the letter a appears (formally the vocabulary for these models contains two predicates: unary P_a and binary $<$). It is known that regular languages are exactly those definable by a MSO formula on this word model [13, Thm 5.2.3]. By Pumping lemma, the language $\{a^k b^k : k \in \mathbb{N}\}$ is not regular, hence not MSO-definable.

From a word represented by $(X, P_a, <)$ we would construct a graph model (X, E) where edges connect possitions of different letters, formally $E(u,v) \iff (P_a(u) \not\Leftrightarrow P_a(v))$. For example for a word *abbaa*, we get the word model $X = [1,5]$ with the usual ordering $<$, and the predicate $P_a = \{1,4,5\}$. The associated graph model is the complete bipartite graph between sets $\{1,4,5\}$ and $\{2,3\}$ given as X and the adjacency relation E.

If the existence of a Hamiltonian cycle in a graph would be decided by an MSO_1 formula, then we replace each occurrence of the adjacency predicate $E(u,v)$ by the the equivalent expression $P_a(u) \not\Leftrightarrow P_a(v)$. The resulting formula together with an expression that all b's follow all a's, namely $\forall u, v : (P_a(u) \wedge \neg P_a(v)) \Rightarrow u < v$, would yield an MSO formula that defines $\{a^k b^k : k \in \mathbb{N}\}$, a contradiction.

Exercise 51: Specify axioms for the theories of simple (i.e. loopless) undirected graphs in MSO_1 and MSO_2.

Exercise 52: Show that any MSO_2 expression without edge set variables/constants can be transformed to an equivalent MSO_1 expression and vice versa:

Exercise 53: Show that in MSO_2 only the $\in$ predicate suffices to substitute the other predicates. Show that the predicates for set equality and set inclusion could be implemented in this way.

5.4 Alhorithmic metatheorem for bounded treewidth

We show that MSO_2 properties can be tested in linear time on bounded treewidth graphs. We avoid the word "efficiently" in this context since the evaluation time is a tower function whose height is proportional to the quantifier depth of the given formula.

Theorem 5.2 (Courcelle [10])**.** *Let $\mathcal{G}$ be a class of graphs of treewidth bounded by k. Then every MSO_2 formula Φ can be evaluated on every graph $G \in \mathcal{G}$ in time $c|V_G|$, where the constant c depends only on k and the formula Φ.*

In the first step we reduce the expressive power of the formula.

Proposition 5.3. *For every MSO_2 formula Φ exists an MSO_1 formula Φ' s.t. on any graph G there exists a graph G' of the same treewidth such that $G \models \Phi$ if and only if $G' \models \Phi'$*

Proof. Subdivide each edge of G once to get G', by Proposition 3.4 $\mathrm{tw}(G') = \mathrm{tw}(G)$. The new vertices correspond to the edges of G.

To get Φ' we first rewrite the incidence relation by the adjacency relation. As the incidence relation is not symmetric as the adjacency relation we shall distinguish the vertices from edges by an extra predicate P, which shall be valid on all variables that correspond to edges in Φ. This could assured when P holds only on vertices of degree two, and when P is satisfied on exactly one of each pair of adjacent vertices of G'.

The predicate P may wrongly identify the added vertices only on components that are cycles. But this situation is isomorphic (in G') to the correct identfication of the added vertices and makes no impact on the evaluation of Φ'.

Observe that the properties of P can be expressed by a subformula with three quantifiers. Hence the quantifier depth of Φ' is the maximum of four, and the quantifier depth of Φ increased by one. □

The following approach to the proof of Courcelle's theorem is due to Hliněný, Král' and Obdržálek[3]. We first derive several properties of evaluation of MSO_1 formuli on general graphs.

Assume that Φ is in the prenex normal form and let $\Psi(z_1, \ldots, z_q)$ be the quantifier-free part of Φ. We distinguish between q_v variables that are quantified over single vertices and call these *individual variables* and the remaining q_s variables that are quantified over sets of vertices, called *set variables*. The total quantifier depth is $q = q_v + q_s$.

Observation 5.4. *When a set assigned to a variable z_i contains a vertex u that is not assigned to any other individual variable, then u does not participate in any of the membership predicates. Hence the evaluation of Ψ is independent of its presence in z_i.*

Hence, we may assume without loss of generality that during evaluation of Ψ, the set variables are restricted only to those vertices that are assigned to individual variables. Consequently, $G \models \Psi(z_1, \ldots, z_q) \iff G' \models \Psi(z'_1, \ldots, z'_q)$, where G' is the subgraph of G induced by the vertices assigned to individual variables $z_i = z'_i$, and where each set variable z'_i is restricted to this subgraph.

The evaluation of Φ on a general graph G on n vertices can be represented as a rooted tree with the following structure.

Definition. The *evaluation tree* for a formula Φ and a graph G is a rooted tree F with $q + 1$ levels constructed recursively as follows. For each level $l = 1, \ldots, q$, a node w of the l-th level corresponds to a particular choice of the first $l - 1$ quantified variables in the order of quantification. If z_l is an individual variable, then w has n children, representing all possible choices of vertices for z_l. Analogously, for a set variable z_l there are 2^n choices among all possible subsets of V_G.

An example of the evaluation tree for the rather trivial formula for the existence of the set of all vertices and G being a path of length two is depicted in Fig. 19. In this figure — and we assume this also in general — edges between w and its children are labeled by the chosen values of the associated variable.

The answer whether Φ is satisfied on G can be obtained by a traverse through F from leaves towards the root. Leaves in F correspond to different assignments of Ψ, and can be evaluated directly. The value of each internal node straightforwardly depends on the quantifier and values of its children: a node w corresponding to a universally quantified variable ($\forall$) must have all children satisfied to became satisfied as well. Similarly, an existentionally quantified variable ($\exists$) needs for the corresponding node at least one satisfied child.

[3]personal communication to D. Král', 2011

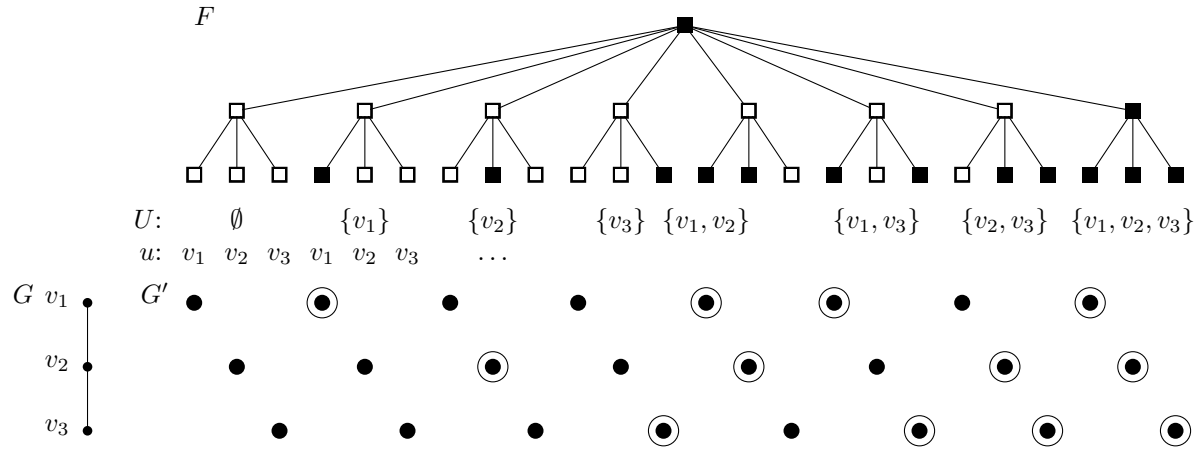


Figure 19: The evaluation tree for $\Phi = \exists U \forall u : u \in U$ and $G = P_3$. Under each leaf is depicted the graph G' and the restricted domain of the set variable U. Black boxes of F are truly evaluated vertices, while white are falsely evaluated.

However this approach can be used in general, its disadvantage is that the evaluation tree F is too large. First we show a method that reduce the tree such that size of the resulting tree is bounded by a constant independent of the size of G. Later we show, that the reduced trees can be computed effectively on graphs of bounded treewidth.

The reduction goes inductively in the bottom-up manner. Each leaf of F corresponds to an evaluation of Ψ on a subgraph G', determined by q_v vertices. The number of nonisomorphic graphs with at most q_v vertices is bounded by $2^{\binom{q_v}{2}} = 2^{O(q_v^2)}$.

The number of possible assignments of individual and set variables over G' is bounded by $q_v^{q_v}\ (2^{q_v})^{q_s} = 2^{O(q_v(q_s+\log q_v))}$. In total, we get $2^{O(q_v q)}$ nonisomorphic combinations of G' and variable assignments.

Leaves of F correspond to such combinations, hence there are $2^{O(q_v q)}$ isomorphism classes for the leaves. Assume that w is a parent of several leaf vertices. As leaves from the same isomorphism class have the same value, the evaluation of w does not affect when leaves from some particular isomorphism class appear once or more times among its children. It suffices to keep one leaf for each isomorphism class that appears there. Hence, we may remove redundant children of w and do the same for all vertices at the q-th level of F. After this adjustment, each node at the q-th level of F will have at most $2^{O(q_v q)}$ children — leaves.

We now distinguish nodes at the q-th level by their children. There are $2^{2^{O(q_v q)}}$ possible sets of children. Each such set defines the isomorphism class of a tree of height one. We determine isomorphism classes of all subtrees rooted in vertices at the q-th level. Now we reduce isomorphic subtrees under a common parent analogously as we have removed redundant leaves.

In general, assume that each node on the $(l+1)$-th level belongs to one of the possible t isomorphism classes. Then, if a node on l-th level has two isomorphic children, we remove one of them and reduce the tree as along as possible. Then the isomorphism class of a node

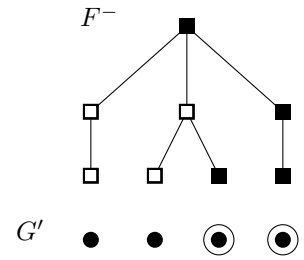


Figure 20: The reduced tree for the tree from Fig. 19.

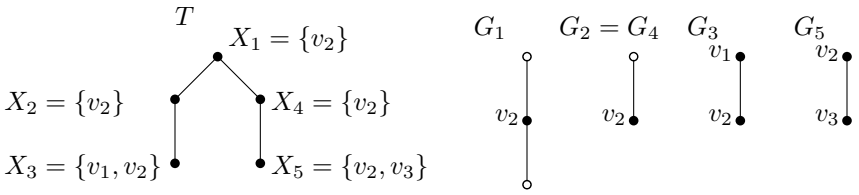


Figure 21: A tree decomposition of $G = P_3$ and the corresponding graphs G_i.

on the l-th level is defined as the set of isomophism classes of its children. Consequently, the number of isomorphism classes at the l-th level is bounded by 2^t.

Definition. The tree F^- obtained by the above described reduction toward the root of the evaluation tree F is called *reduced evaluation tree* for Φ on G.

The following two claims are immediate:

Observation 5.5. *The reduced evaluation tree F^- can be evaluated instead of the expression tree F to decide whether $G \models \Phi$ or not.*

Observation 5.6. *The size of the reduced evaluation tree F^- is bounded by $\left.2^{\cdot^{\cdot^{\cdot^{2^{O(q_v q)}}}}}\right\}_{q+1}$. In particular, this bound depends only on the number of individual and set variables and is independent on the size of the graph G.*

The reduced tree F^- for the tree from Fig. 19 is depicted in Fig. 20

So far we did not utilize the fact that G has bounded treewidth. We show how this property enables us to construct the reduced expression tree for a node of the tree decomposition from the evaluation trees of its children. For this purpose we first generalize the notion of the evaluation tree and the associated configuration.

Proof of Theorem 5.2. Let X_i be a node of a tree decomposition of G and let G_i be the subgraph induced by X_i and its descendants. (See Fig. 21.)

We extend the evaluation tree F_i for Φ on G_i to adopt the situation where value of some individual variable does not belong to G_i: for each individual variable z_l we add to each associated node w an extra child representing the case where z_l is not determined. We say that its value is *external* and label the edge by "ext". The resulting tree is called *partial evaluation tree.*

Note that partial evaluation trees cannot be evaluated directly, since predicates on external values may not evaluated and hence the value of Ψ is not well defined in some leaf configurations. On the other hand, if we prune in such F_i all configurations containing a external choices, then Ψ is well defined on each leaf. Observe that the resulting evaluation tree tests whether $G_i \models \Phi$.

We reduce the size of partial evaluation trees similarly to the case of general graphs.

A leaf of F_i corresponds to a situation when some or all individual variables z_i are assigned the elements of G_i and set variables z_i are assigned appropriate subsets of the individual variables. Each such assignment again defines a subgraph of G_i. In comparison to the case of general graphs we distinguish which elements of $G[X_i]$ are selected by the assignment, hence we include the whole subgraph $G[X_i]$ in the resulting subgraph G', and extend all the choices of set variables on this domain.

We again list all nonisomorphic graphs G' together with the assignment and require that the possible isomorphism must fix $G[X_i]$. Such representative of the isomorphism class will be called a *configuration*. We summarize that a configuration is formed of

- the graph G' induced by the union of $G[X_i]$ and at most q_v further vertices,
- a partial mapping between the individual variables and the vertices of G',
- a mapping between the set variables and the subsets of $V_{G'}$.

Namely, in a configuration a vertex variable may be assigned a particular vertex of X_i or it could be assigned an unspecified vertex in $V_{G'} \setminus X_i$ (we call it an *internal* value) or it may not be assigned any value. Set variables are composed from the elements of $G[X_i]$ and chosen internal individual variables.

We first reduce labels of the edges in the tree F_i so that whenever an individual variable is assigned an element of $G_i \setminus G[X_i]$ the label becomes "int", and so that labels representing choices of set variables became restricted to $G[X_i]$. Still, one node w of F_i may leave several edges with the same label. Now we are ready transform the tree F_i into the *partial reduced evaluation tree* F_i^- by the same procedure as in the case of general graphs.

The number of nonisomorphic configurations is at most $2^{\binom{q_v+k+1}{2}}(q_v+k+2)^{q_v}(2^{q_v+k+1})^{q_s} = 2^{O((q_v+k)(q+k))}$. By the same arguments as before we get that each node X_i may be assigned a the partial reduced evaluation tree F_i^- of size $\left.2^{\cdot^{\cdot^{\cdot^{2^{O((q_v+k)(q+k))}}}}}\right\} q+1$.

The direct evaluation of F_i^- makes no sense similarly as for F_i. On the other hand, the removal of all branches with choices of external values from F_i^- yields an evaluation tree for $G_i \models \Phi$ as it has been observed for F_i.

It remains to show, how F_i^- can be constructed without F_i, but with help of the tree decomposition T of G. Without loss of generality we assume that the tree decomposition T is nice.

In each leaf X_i of T the tree F_i^- can be constructed directly by the reduction of the full evaluation tree, since the size of G_i is bounded by a constant $O(k^2)$ (encounting edges, for an example see Fig. 22 left).

For a forget node X_i above X_j we modify each configuration in F_j^- such that the forgotten vertex become internal whenever is chosen for an individual variable. We restrict all choices of set variables to $G[X_i]$. Consequently, we reduce the evaluation tree as some configurations may became isomorphic (see Fig. 22 right).

For an introduce node, each configuration containing an external choice is altered as follows: for each subset of unassigned variables (external) we introduce a new case when variables of this subset are assigned the new element of X_i. Analogously, set variables are extended in all possible combinations by the new element. These new choices result in new branches in F_i^-.

It remains to show how a tree F_i^- can be obtained from trees F_j^- and F_k^- if the node X_i is the parent of nodes X_j and X_k. We use a specific product of trees defined as follows. We choose the root of F_i^- as the pair formed from the two roots of F_j^- and F_k^-. By induction assume that $w_i = (w_j, w_k)$ is a node of F_i^- on the l-th level where $w_j \in F_j^-$ and $w_k \in F_k^-$. For each combination of children w'_j and w'_k of w_j and w_k, respectively, we put $w'_i = (w'_j, w'_k)$ in F_i^- as a child of w_i if the following condition is satisfied:

- If the l-th quantified variable in Φ is an individual variable $z_{l'}$ then
  - either z_l is assigned the same vertex of X_i in both w'_j and w'_k (then in w'_i the variable z_l keeps the same value),
  - or in w'_j the variable z_l external and in w'_k it is an internal element of $G_k \setminus G[X_k]$
  - or vice-versa where k and k' are exchanged with j and j'
    (in both cases z_l becomes in w_i an internal element of $G_i \setminus X_i$)
  - or the variable z_l chosen external in both w'_j and w'_k; here z_l remains external also for w_i.
- If z_l is a set variable then
  - the value of z_l on w'_j, on w'_k and hence also on w'_i restricted to X_i is the same in all three cases. I.e. in all three trees the labels of edges towards w'_j, w'_k and w'_i are the same.

The value of a set variable z_l is determined on each leaf of F_i^- as the union of its values in the corresponding leaves of trees F_j^- and F_k^-. Note that it may contain futher internal vertices of G_j or of G_k. These internal vertices must be chosen by individual variables $z_{l'}$ including also the cases $l' > l$, i.e. they might be on a higher depth of the quantification.

When the entire product tree is formed, we reduce it again.

Observe that in the tree F_i^- any possible configuration on G_i provides particular configurations in trees F_j^- and F_k^-. Also we have just described a method that allows us to reconstruct the original configuration from the two derived ones.

The time complexity is mainly determined by the construction of the tree products for the union nodes, which has $\left(2^{\cdot^{\cdot^{2^{O((q_v+k)(q+k))}}}}\right\}q+1\right)^2 = 2^{\cdot^{\cdot^{2^{O((q_v+k)(q+k))}}}}\right\}q+1$ vertices.

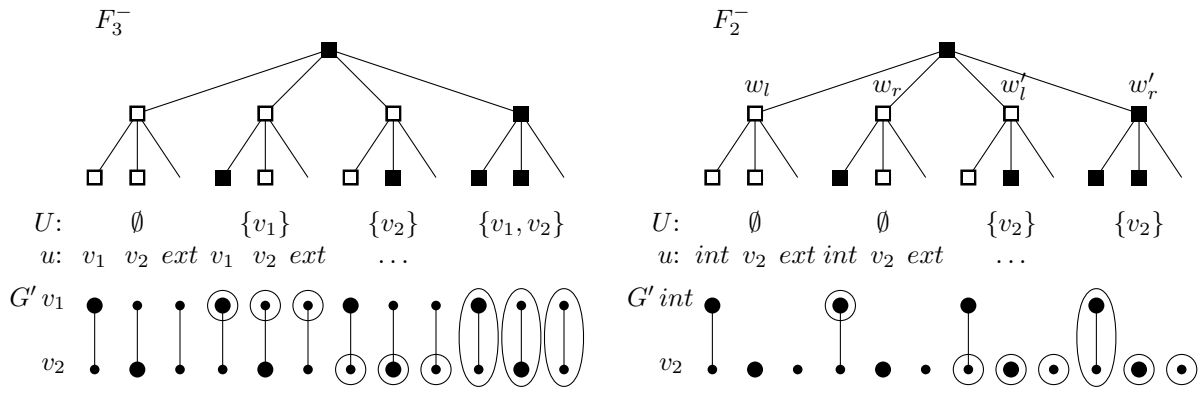


Figure 22: Partial reduced evaluation trees for the decomposition from Fig. 21 — leaf and forget nodes. Bold elements in graphs are assigned to some variables, while thin are the remaining elements of $G[X_i]$. Nodes corresponding to the external choices are excluded from the evaluation.

We conclude that the total time complexity of this method is linear in n, the number of vertices of G, with the multiplicative constant of order $2^{\cdot^{\cdot^{\cdot^{2^{O((q_v+k)(q+k))}}}}}\Big\}q+1$. □

Exercise 54: Decide whether the problem $\chi(G^2) \leq k$ is expressible in MSO_1 or MSO_2 and if so, construct a linear-time algorithm for graphs of bounded treewidth.

Exercise 55: Settle the computational complexity of the decision problem whether a graph of bounded treewidth is 3-choosable, provided the list assignment is a part of the input. Could this property be expressed in MSO_1 or in MSO_2?

6 Algorithms based on other width parameters

6.1 Two algorithms for graphs with bounded NLC-width

We first present a *ch*-coloring algorithm for graphs of bounded NLC-width. We use an analogous notation as for treewidth-bounded graphs: X_l stands for a node of the expression tree and G_l is the subgraph of G induced by vertices of the introduce nodes which are descendants of X_l. Finally denote $E^{i,j}(G_l)$ the set of edges of G_l between vertices of labels i and j.

Consider a coloring c of a graph G_i, and let W be a color class of c, i.e. a set of vertices of G_i of the same color. We say that W is labeled by $J \subseteq I$, if J is exactly the set of labels used on W, i.e., if every label of J appears on some vertex of W and no vertex of W has a label outside J. (Vertices outside W may or may not have labels of J.)

We again involve dynamic programming and store a table $\mathtt{Tab}_l$ in each node X_l of the expression tree. Each element $\mathbf{x} \in \mathtt{Tab}_l$ is an integer vector of length $2^{|I|}-1$ and describes a possible coloring c of G_l. The entries of $\mathbf{x}$ are indexed by nonempty subsets $J \subseteq I$ and the

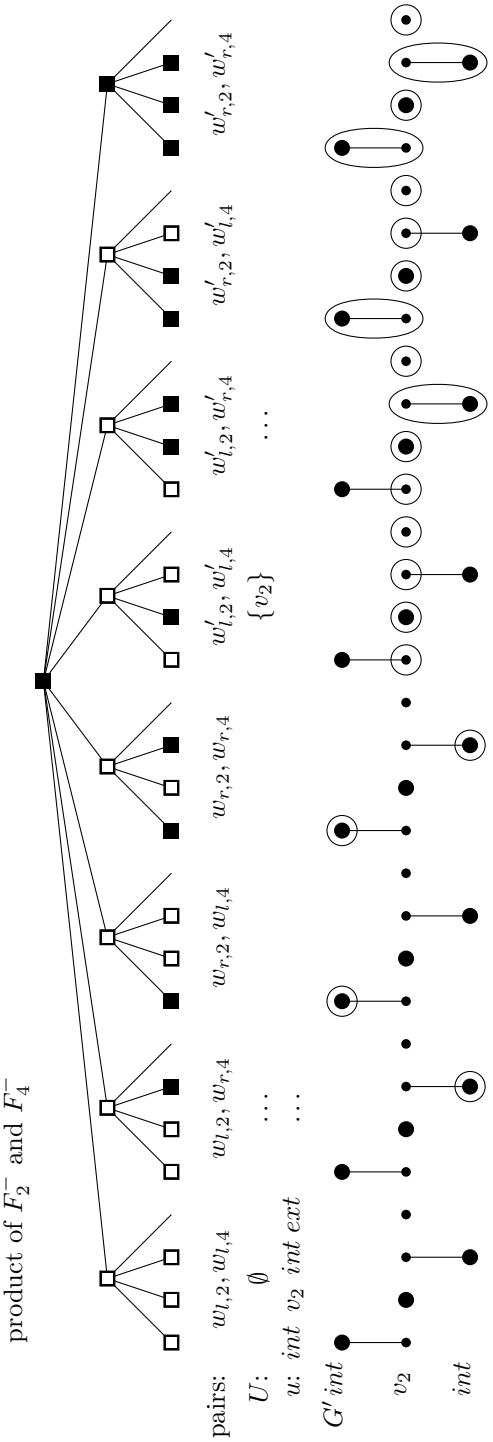


Figure 23: Product tree formed for the union node.

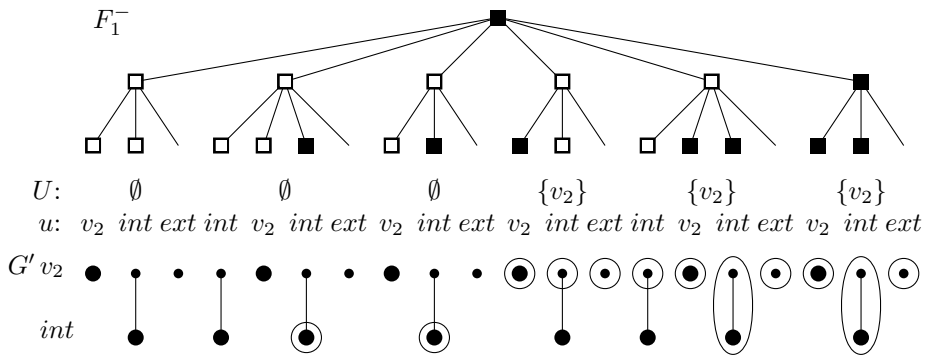


Figure 24: Partial reduced evaluation tree for the union node obtained from the ree from Fig. 23. Bold edges indicate the branches without external values. When we forget labels of elements (e.g. make all elements internal) we get the reduced evaluation tree of Fig. 20. In our example the two trees are indeed isomorphic.

value of $\mathbf{x}_J$ is the number of color classes of c which are labeled by J. Note that multiple colorings may correspond to the same vector $\mathbf{x}$.

Observe that the table size is bounded by $\binom{2^k+ch-1}{ch}$, where $k = |I|$, since the sum of coordinates of each vector of Tab_l is at most ch (and the number above is the number of combinations to distribute at most ch ones into $2^k - 1$ slots).

The algorithm for evaluating $\texttt{Tab}_l$ proceeds as follows: At relabel nodes ρ_r we have to update the entries for the color classes which contain vertices labeled by i and convert them to those which have $r(i)$ instead of i in the index set J.

The most complex is the computation of the table $\texttt{Tab}_l$ for a join node $X_l = \oplus_S$. Assume that a color class of G_l with vertices labeled by J splits into two classes of G_m and of $G_{m'}$ with label sets K and K' (at least one of these two classes is nonempty). Then the variable $a_{K,K'}$ describes how many times such a situation appears. Each solution of the system of inequalities describes how color classes of G_m and of $G_{m'}$ can be matched together to form a feasible coloring of G_l, and vice versa. Observe that the last condition tests whether the vector $\mathbf{x}$ describes a coloring with at most ch colors.

The time complexity is asymptotically determined by solving the system of inequalities which is done for exactly $n - 1$ union nodes. We have to test each of $\binom{2^k+ch-1}{ch}$ possible solutions $\mathbf{x}$. The feasibility of the system can be tested by brute evaluation of all possible assignments for variables $a_{K,K'}$ in time $O\left(\binom{4^k+ch-1}{ch}\right)$, since the total number of variables $a_{K,K'}$ is $4^k - 1$.

The overall time complexity is $O\left(\binom{4^k+ch-1}{ch}4^k n\right)$.

ch-COLORABILITY
Input: *An expression tree E of a graph G of fixed NLC-width $k = |I|$.*
Output: *Decision whether $\chi(G) \leq ch$, for a fixed ch.*
begin
  mark all nodes as unfinished
  for all leaves $X_l = i(v)$ store in $\mathtt{Tab}_l$ the unique vector $\mathbf{x}$:
    $\mathbf{x}_{\{i\}} = 1$ and $\mathbf{x}_J = 0$ for all $J \neq \{i\}$ and mark X_l as finished
  while exists an unfinished node X_i with all children finished **do**
    begin
      case X_l of
        — *relabel node* ρ_r with child X_m
          then for every $\mathbf{y} \in \mathtt{Tab}_m$ store in $\mathtt{Tab}_l$ the vector $\mathbf{x}$:
            $\mathbf{x}_J = \sum_{J':r(J')=J} \mathbf{y}_{J'}$
        — *union node* $\oplus_S$ with children X_m, $X_{m'}$
          then for each choice of $a_{K,K'}$ where $K, K' \subseteq I, K \neq \emptyset \vee K' \neq \emptyset$:
          such that $\sum_{K,K'} a_{K,K'} \leq ch$ and $a_{K,K'} = 0$ whenever $(K \times K') \cap S \neq \emptyset$:
            $\forall K \neq \emptyset, K \subseteq I : \mathbf{y}_K := \sum_{K' \subseteq I} a_{K,K'}$
            $\forall K' \neq \emptyset, K' \subseteq I : \mathbf{y}'_{K'} := \sum_{K \subseteq I} a_{K,K'}$
            $\forall J \neq \emptyset, J \subseteq I : \mathbf{x}_J := \sum_{K \cup K' = J} a_{K,K'}$
            if $\mathbf{y} \in \mathtt{Tab}_m$ and $\mathbf{y}' \in \mathtt{Tab}_{m'}$ **then** store $\mathbf{x}$ in $\mathtt{Tab}_l$
        esac
      mark X_l as finished
    end;
  answer "YES" if $\mathtt{Tab}_{root} \neq \emptyset$ and "NO" otherwise.
end.

This linear-time decision algorithm for fixed number of colors ch can be used for the case when ch is a part of the input, or with a litle modification to the optimization algorithm. Observe that the tables $\mathtt{Tab}_i$ may contain at most $(n+1)^{2^k}$ distinct entries, since we may restrict ourselves to the case when $ch \leq n$. For fixed k this yields a polynomial bound on the sizes of the tables as well as on the time complexity. Note that the assumption that ch is fixed is essential to express a ch-coloring in MSO_1, so we get beyond the algorithmic meta-theorem for graphs of bounded cliquewidth that is discussed in the next section.

We have already argued that the existence of a Hamiltonian cycle can be expressed by MSO_2 but not MSO_1 formula. Hence this problem is not covered by Theorem 6.1.

We now give a hint that this problem is solvable in polynomial time on graphs of bounded NLC-width [41]. The key ingredient is the description how a Hamiltonian cycle of G intersects the subgraph of G corresponding to a node X_l of the expression tree. The intersection is a collection of paths. Internal nodes of these paths do not affect further compositions of the parts of the cycle together. Hence it suffices to record only information about the ends of the paths in the intersection. In addition, vertices of the same labels are treated equally in the further composition of the graph, so it suffices to record only pairs of labels of the ends of the paths.

Exercise 56: Provide details of the algorithm for Hamiltonian cycle in graphs of bounded NLC-width, in particular how nodes of the expression tree are processed.

Exercise 57: For fixed d, design an algorithm that decides the existence of a d-regular subgraph in a graph of bounded NLC-width.

6.2 Algorithmic metatheorem for bounded NLC-width

Theorem 6.1 (Courcelle, Makowsky and Rotics [11]). *Let $\mathcal{G}$ be a class of graphs of NLC-width bounded by k. Then every MSO_1 formula Φ can be evaluated on every graph $G \in \mathcal{G}$ in time $2^{\cdot^{\cdot^{2^{O(q(q+\log k))}}}}\Big\}q+1 |V_G|$, where q is the number of quantifiers of the formula Φ, provided that the expression tree of G of width at most k is given.*

Idea of the proof. We may use the same method as for the proof of Theorem 5.2. We gradually build reduced evaluation tree F_i^- for each node X_i of the expression tree E.

Each configuration associated to a leaf node consists of

- a graph G' on at most q_v vertices; this is a subgraph of G_i corresponding to the the graph formed by the expression tree rooted X_l, in particular these vertices are labeled,
- a partial surjective mapping between the individual variables and vertices of G',
- a mapping between the set variables and subsets of $V_{G'}$.

The partial assignment is represented in F by use of external labels. Note that in contrast with the former proof there are no internal labels.

We consider two configuration isomorphic if the the graph homomorphisms respects also all three associated mappings: vertex labels and both variable assignments.

Since the the number of distinct configurations is bounded by $2^{\binom{q_v}{2}}(q_v+1)^{q_v}(2^{q_v})^{q_s}k^{q_v} = 2^{O(q_v(q+\log k))}$, the size of the reduced tree is bounded by $2^{\cdot^{\cdot^{2^{O(q_v(q+\log k))}}}}\Big\}q+1$.

It remains to argue how the reduced evaluation tree F_l^- is formed from such trees associated with the children of X_l.

If $X_l = i(v)$ is a leaf then $F_l = F_l^-$ is constructed directly. Observe that it is a full binary tree — for each individual variable there are two choices (v or ext) as well as for set variables ($\{v\}$ or $\emptyset$).

If $X_l = \rho_r(X_m)$ then we apply the relabeling on each configuration on a leaf of F_m^- and reduce the resulting tree immediately.

If $X_l = \oplus_S(X_m, X_{m'})$ then we first construct an analogous product of trees F_m^- and $F_{m'}^-$. This product is reduced afterwards.

In the product two cases may appear for individual variables: the assignment of a particular vertex may be combined with the external choice, or two external choices may be combined together. For the set variables we simply perform the union of the associated sets.

Observe that the graph associated to any resulting configuration can be split in two graphs: one corresponding to the configuration of F_m^-, the other of $F_{m'}^-$. The edges between these two parts are detemined by the relation S and the vertex labels. □

6.3 Bounded neighborhood diversity

Definition. The *neighborhood partition* is the equivalence relation, where two vertices u and v are in the same class if $N(u) \setminus v = N(v) \setminus u$

An equivalece class of a neghborhood partition is called a *neighborhood class.*

The *neighborhood diversity* $\mathrm{nd}(G)$ is the number of classes of the coarsest neighborhood partition.

Observe that a graph G has neighborhood diversity bounded by k if its vertces can be labeled by set $I = \{1, \ldots, k\}$, such that is some pair of vertices are adjacent, then all pairs with the same labels are adjacent as well.

A graph of bounded neighborhood diversity can be described upto an isomorphisms by the sizes of each class of the neighborhood partition as a mapping $I \to \mathbb{N}$ and by a symmetric binary relation S on the set I.

Exercise 58: Relate the neighborhood diversity to the NLC-width.

Graphs of bounded neighborhood diversity are closed on taking distance powers. The t-th *distance power* G^t of G is obtained from G by adding further edges between distinct vertices that are in G at distance at most t.

Observation 6.2. *For any t and any graph it holds that* $\mathrm{nd}(G^t) \leq \mathrm{nd}(G)$.

As a corollary we obtain that the *ch*-coloring algorithm for graphs of bounded NLC-width can be adapted straightforwardly to an *ch*-coloring algorithm for distance powers of graphs of bounded neighborhood diversity.

This result can be indeed strenghtened. Todinca and Suchan showed that the NLC-width can be bounded when taking distance powers [38]. In the same paper they also provided as hoc coloring algorithm with better performance than the coloring algorithm using the obtained bound on $\mathrm{nlcw}(G^t)$.

We now present an algorithmic metatheorem by Lampis who showed that MSO_1 graph properties can be decided on graphs of bounded neighborhood diversity in constant time, where the constant is only doubly exponentional in the quantifier depth. Compare with the tower function of the Theorem 5.2, specified in Observation 5.6.

Theorem 6.3 (Lampis [19])**.** *Let G be a graph of bounded neighborhood diversity and Φ be a MSO_1 formula with q_v individual and q_s set variables. Then, given a neighborhood partition of G with* $\mathrm{nd}(G)$ *classes, there exists an algorithm which can decide whether $G \models \Phi$ in time $2^{O(2^{q_s} q_v \mathrm{nd}(G) + q_v \log q_v)}$.*

The essential argument is that to $G \models \Phi$ if and only if its subgraph $G' \models \Phi$, where the size each neighborhood class is at most $2^{q_s} q_v$.

Proof. Consider that Φ is evaluated on on G. We construct the evaluation tree, F for G. We call nodes of the evaluation tree *configurations*. At a level i, each configuration of corresponds to the assignment $\mathcal{Z} = (Z_1, \ldots, Z_i)$ of variables $z_1, \ldots, z_i$.

Without loss of generality we may consider only nonisomorphic configurations. By isomorphic configurations we mean those assignements $\mathcal{Z}$ for which G has an isomorphism that set-wise fixes each $Z_k \in \mathcal{Z}$ and also each neighborhood class. Hence we may view $\mathcal{Z}$ as a refinement of the neighborhood partition, formally as a relation $\sim_{\mathcal{Z}}$ on V_G where $u \sim_{\mathcal{Z}} v$ if and only if $(N(u) \setminus v = N(v) \setminus u) \wedge (\forall Z \in \mathcal{Z} : u \in Z \iff v \in Z)$. Observe that the nonisomorphic configurations $\mathcal{Z}$ can be described by the sizes of equivalence classes of $\sim_{\mathcal{Z}}$.

The equivalence classes of $\sim_Z$ are of two categories. Either it contains a vertex v chosen for some individual variable z_k, i.e. $Z_k = \{u\}$. In this case u solely forms an equivalence class of $\sim_Z$. Alternatively, the class contains only vertices that have not been chosen by any individual variables. Let's call these *free classes*.

With a slight abuse of notation let $\Phi(\mathcal{Z})$ be the subformula of Φ where variables $z_1, \ldots, z_i$ have fixed values $\mathcal{Z} = (Z_1, \ldots, Z_i)$. We iteratively reduce the decision tree F into F^-, s.t. for each configuration $(G, \mathcal{Z})$ of F we find a subgraph G' of G and a reduced configuration $(G', \mathcal{Z}')$ satisfying that $G \models \Phi(\mathcal{Z}) \iff G' \models \Phi(\mathcal{Z}')$.

If $(G, \mathcal{Z})$ is a leaf of F, let $\mathcal{Z}'$ be the restriction of $\mathcal{Z}$ to the vertices chosen by individual variables. In other words, $\sim_{\mathcal{Z}'}$ consists of the non-free classes of $\sim_{\mathcal{Z}}$. Let G' be the subgraph of G induced the vertices of $\bigcup \mathcal{Z}'$. As the vertices of the free classes of $\sim_{\mathcal{Z}}$ does not participate in any of the predicates of Φ, they do not affect validity of $\Phi(\mathcal{Z})$ (which already became quantifier free). Hence $G \models \Phi(\mathcal{Z}) \iff G' \models \Phi(\mathcal{Z}')$. We put the resulting configuration $(G', \mathcal{Z}')$ an independent vertex in the forest F^-.

Consider a configuration $(G, \mathcal{Z})$, where $\mathcal{Z} = (Z_1, \ldots, Z_{i-1})$. By induction assume that $(G, \mathcal{Z})$ is in F the parent of configurations $(G, \mathcal{Z}, Z_i^1), \ldots, (G, \mathcal{Z}, Z_i^t)$ where for each $Z_i^j, j \in \{1, \ldots, t\}$ there exists a reduced configuration $\left(G^{j\prime}, \mathcal{Z}^{j\prime}, Z_i^{j\prime}\right)$ where:

- $G^{j\prime}$ is a subgraph of G,
- each set of the set system $\mathcal{Z}^{j\prime} \cup \{Z_i^{j\prime}\}$ is the restriction of the corresponding set from $\mathcal{Z} \cup \{Z_i^1\}$ to $V_{G^{j\prime}}$, and
- the reduced configuration satisfies $G \models \Phi(\mathcal{Z}, Z_i^j) \iff G^{j\prime} \models \Phi\left(\mathcal{Z}^{j\prime}, Z_i^{j\prime}\right)$.

We construct $(\mathcal{G}', \mathcal{Z}')$ where each class C of $\sim_{\mathcal{Z}'}$ has size $\max\{|C^j|, j \in \{1, \ldots, t\}\}$, where C^j is the class corresponding to C in the partition $\sim_{\mathcal{Z}^{j\prime}}$. As each $Z_k^{j\prime} \in \mathcal{Z}^{j\prime}$ is a restriction of $Z_k \in \mathcal{Z}$ to a subgraph $G^{j\prime}$ of G, the sizes of classes of $\sim_{\mathcal{Z}^{j\prime}}$ are bounded by the sizes of corresponding classes of $\sim_{\mathcal{Z}}$. Hence the resulting partition $\sim_{\mathcal{Z}'}$ has classes bounded by the same upper bounds. Therefore $\mathcal{Z}'$ induces a subgraph G' of G.

We make the resulting configuration $(G', \mathcal{Z}')$ the common parent of configurations $(G, \mathcal{Z}, Z_i^1), \ldots, (G, \mathcal{Z}, Z_i^t)$ in the forest F^-. (For the purposes of further evaluation we may keep only nonisomorphic children in F^-.)

Now, for any assignment Z_i to z_i, consider the configuration $\left(G^{j'}, \mathcal{Z}^{j'}, Z_i^{j'}\right)$ corresponding to $(G, \mathcal{Z}, Z_i)$. By induction $G \models \Phi(\mathcal{Z}, Z_i) \iff G^{j'} \models \Phi\left(\mathcal{Z}^{j'}, Z_i^{j'}\right)$, By the construction of G' it holds that $G^{j'} \subseteq G' \subseteq G$, and in particular $Z_k^{j'} = Z'_k \cap V_{G^{j'}}$ for all $k \in \{1, \ldots, i-1\}$. Hence, we extend $\mathcal{Z}'$ by $Z_i^{j'}$ to conclude $G \models \Phi(\mathcal{Z}, Z_i) \iff G' \models \Phi\left(\mathcal{Z}', Z_i^{j'}\right)$.

Since $G' \subseteq G$, then for any choice of Z'_i in G' we get the configuration $(G, \mathcal{Z}, Z'_i)$ and the corresponding one $(G^{j'}, \mathcal{Z}^{j'}, Z_i^{j'})$. Since $Z_i^{j'} = Z'_i \cap V_{G^{j'}}$ we get that $G' \models \Phi(\mathcal{Z}', Z_i') \iff G^{j'} \models \Phi\left(\mathcal{Z}^{j'}, Z_i^{j'}\right) \iff G \models \Phi(\mathcal{Z}, Z'_i)$.

The last two paragraphs show that $G \models \Phi(\mathcal{Z}) \iff G' \models \Phi(\mathcal{Z}')$ — each children of $(G, \mathcal{Z})$ corresponds to an equivalently evaluated children of $(G', \mathcal{Z}')$ and vice versa. Consequently, F^- can be evaluated instead of F to decide whether $G \models \Phi$.

It remains to calculate the sizes of the graphs in the configurations of F^-. As each class of $\sim_{\mathcal{Z}}$ containing an individual variable is of size one, we focus on free classes only.

We claim: *If the formula $\Phi(\mathcal{Z})$ contains q_v^i individual and q_s^i set quantifiers, then for each reduced configuration $(G', \mathcal{Z}')$ each free class of $\sim_{\mathcal{Z}'}$ contains at most $2^{q_s^i} q_v^i$ vertices.*

The statement is true on leaves of F^-, where $q_v^q = 0$ and all free classes are empty in each such G'.

Consider a free class C of $\sim_{\mathcal{Z}'}$. For any Z'_i, the class C is the union of two classes $C \cap Z'_i$ and $C \setminus Z'_i$ of $\sim_{\mathcal{Z}' \cup \{Z'_i\}}$. When z_i is a set veriable, then $|C| = |C \cap Z'_i| + |C \setminus Z'_i| \leq 2 \cdot 2^{q_s^{i+1}} q_v^{i+1} = 2^{q_s^i} q_v^i$ as $q_v^i = q_v^{i+1}$ and $q_s^i = q_s^{i+1} - 1$. When z_i is an individual variable, then $C \cap Z'_i$ has at most one vertex. Analogously, $|C| \leq |C \setminus Z'_i| + 1 \leq 2^{q_s^{i+1}} q_v^{i+1} + 1 \leq 2^{q_s^i} q_v^i$ as $q_v^i = q_v^{i+1} - 1$ and $q_s^i = q_s^{i+1} \geq 0$.

The root of the tree F^- is the configuration $(G', \emptyset)$. As all classes of $\sim_\emptyset$ are free, and since $\sim_\emptyset$ coincides with the neighborhood partition of G, we get that $|V_{G'}| \leq 2^{q_s} q_v \mathrm{nd}(G)$.

If G' has n vertices then the straightforward brute-force algorithm evaluates whether $G' \models \Phi$ in time $O((2^n)^{q_s} n^{q_v} |\Phi|)$. The combination of the two bounds concludes the proof. □

6.4 Bounded vertex cover

An even tighter measure of a graph than the pathwidth or neighborhood diversity is the notion of vertex cover:

Definition. A *vertex cover* of a graph is a set of vertices $W \subseteq V_G$ such that every edge is incident with at least one vertex of W. The minimum size of a vertex cover is denoted by $\mathrm{vc}(G)$

Exercise 59: Relate the size of the minimum vertex cover to the pathwidth and to the neighborhood diversity.

From the computational complexity point of view it makes sense to study on the class of graphs of bounded vertex cover those problems that are known to be difficult on graphs of bounded treewidth or pathwidth.

An example of such problem is $L(2,1)$-Labeling of a graph. Here the task is to decide whether vertices of a given graph G can be labeled by integers from a given interval, such that labels of adjacent vertices differ by at least two, while vertices with a common neighbor should have labels that differ by at least one. The $L(2,1)$-Labeling problem is known to be NP-complete on series-parallel graphs [14].

We show that an analogous problem of $L(1,1)$-Labeling, equivalent to the problem of coloring the second power of a graph, can be solved in polynomial time for graphs of bounded vertex cover [15]. We will comment later how this approach can be extended to $L(2,1)$-labeling. Observe also that the algorithm has better running time than the coloring algorithm for powers of graphs of bounded neighborhood diversity mentioned in th eprevious section.

The instance is formed both from the graph G as well as from the bound on the number of colors ch. When ch is fixed, then the problem can be expressed by MSO_1, and hence could be decided in linear time on graphs of bounded treewidth.

Assume that a graph G has a vertex cover $W = \{v_1, \ldots, v_k\}$. We partition the remaining vertices into sets $I_{\{1\}}, I_{\{2\}}, \ldots, I_{\{k\}}, I_{\{1,2\}}, \ldots, I_{\{1,\ldots,k\}}$, such that the sets are given as $I_J := \{u : (u, v_i) \in E_G \Leftrightarrow i \in J\}$. Without loss of generality we assume that G is connected, hence every vertex outside W belongs to a unique set I_J.

Observe that in any coloring of G^2, vertices of the same set I_J must be given different colors. On the other hand, the colors inside any set I_J can be permuted arbitrarily, since all vertices in I_J have the same neighborhood in W.

We say that a color $C \subseteq V_G$ *appears on* $K \subseteq \mathcal{P}(\{1, \ldots, k\}) \setminus \emptyset$ the sets I_J contain a vertex colored by C if and only if $J \in K$.

Observe that no color can be used on a K containing two index sets J, J' with an nonempty intersection, since vertices from I_J and $I_{J'}$ are at distance two in G.

The core task of the algorithm will be to determine values of variables x_K that describe the desired coloring in the following way: x_K is the number of color classes that appear on K.

G^2-Colorability
Input: *A graph G with vertex cover W of fixed size k, an integer ch.*
Output: *Decision whether $\chi(G^2) \leq ch$.*
begin
  if $ck \leq k$ then use MSO_1
  determine all sets $I_J, J \in \mathcal{P}(\{1,\ldots,k\}) \setminus \emptyset$
  for every partial k-coloring c of G^2 where whole W is colored **do**
    begin
      for every $K \subseteq \mathcal{P}(\{1,\ldots,k\}) \setminus \emptyset$ determine a_K
        to be the number of color classes of c used on K
      solve the following system of inequalities in integer variables x_K
        $\sum_K x_K \leq ch$
        $\forall K : x_K \geq a_K$
        $\forall J \in \mathcal{P}(\{1,\ldots,k\}) \setminus \emptyset : \sum_{K:J\in K} x_K = |I_J|$
        $\forall K$ s.t. $\exists J, J' \in K : J \cap J' \neq \emptyset : x_K = 0$.
      if the system has a solution **then** answer "YES" and **exit**
    end
  answer "NO"
end.

The correctness of the algorithm follows from two facts: If G^2 allows an coloring c', then we can permute colors s.t. the first k colors are used on W. Then for c being the first k colors of c' the system of inequalities has a feasible solution with x_K describing how many color classes of c' are used on K.

In the opposite way, the remaining $ch - k$ color classes are uniquely described by the solution of the system of inequalities. For every K we take $x_K - a_K$ new color classes, each containing a unique vertex from each set I_J where $J \in K$. This provides a valid coloring of G^2. The first inequality yields that the total number of colors is at most ch, the second that it is a valid extension of c, the third that all vertices are colored, and the last condition is necessary and sufficient to get a valid coloring of G^2.

The time complexity is determined by three factors. Firstly, the partition into sets I_J needs kn time, then the number of colorings c which is $O(n^{k+1})$ Secondly, as the time needed to solve the system of inequalities with p integer variables is $O(p^{2.5p+o(p)}L)$, where L is the number of bits needed to encode the input. The first result in this direction was shown by Lenstra in 1983 and later improved several times [22, 1]. As we have 2^{2^k-1} variables, the time needed to resolve the system of inequalities is $O(kn + 2^{2^{2^{k+1}}} \log n)$.

The above algorithm can be extended to $L(p,1)$-labelings for $p \geq 2$ due to the following claim: It possible to assume without loss of generality that labels used on the set W can be shifted to the first $2pk$ or the last $2pk$ labels. Hence, we shall try all such labelings of the set W, and test its possible extension to sets I_J by an analogous system of inequalities.

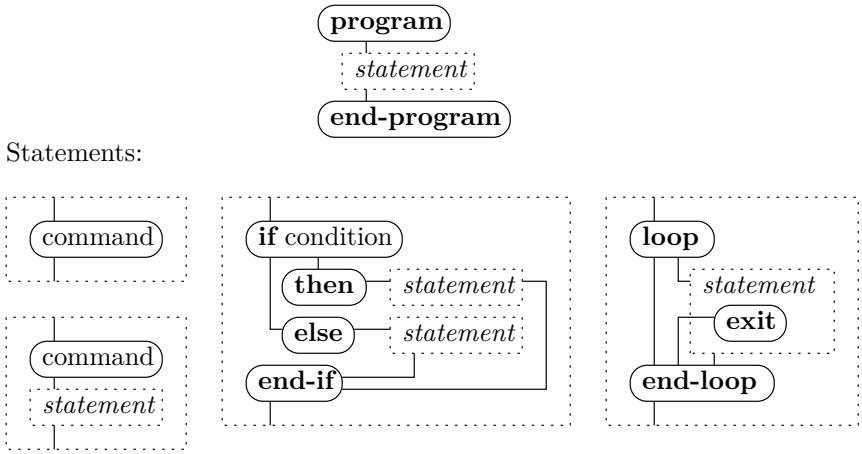


Figure 25: Program constructs and control flow diagrams.

7 Other applications of treewidth

7.1 Treewidth of flow control graphs and register allocation

As a practical application of the notion of treewidth we reproduce here a result on flow control graphs and register allocation [40].

Consider a simple programming language with **if** − **then** − **else** − **end-if** and **loop** − **exit** − **end-loop** constructs.

Roughly speaking, the control flow graph models the possible order of processing statements, i.e. those statements which might be processed contiguously are joined by a (directed) edge. For our purposes we assume that the program constructs and segments of the corresponding control flow graph are defined by the diagrams depicted in Fig. 25.

It is not hard to see that without the keyword **exit** the control flow graph of any program in our language is a series-parallel graph, hence of treewidth 2. The keyword **exit** deserves further explanation: It may appear within any statement inside the **loop** − **end-loop** cycle in a position of a command. In fact more **exit**s may appear there, but each passes the computation to the closest subsequent **end-loop**, as indicated in the figure.

Exercise 60: Prove that if on the program the keyword **exit** appears at most once, then the treewidth of the control flow graph is at most three.

In fact a stronger statement holds.

Proposition 7.1. *The treewidth of the control flow graph for any program written in our simple language is bounded by 3.*

Exercise 61: Prove Proposition 7.1.

For widely used structured programming languages, the following result was obtained in [40]:

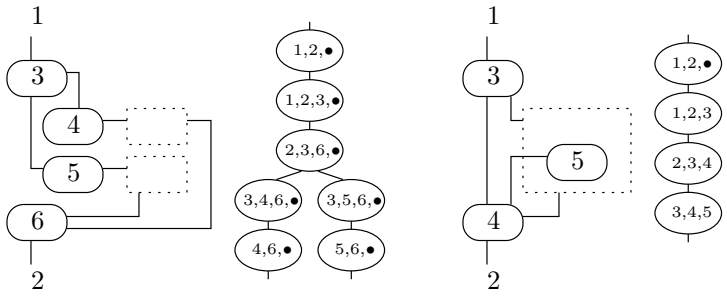


Figure 26: From control flow graph to tree decomposition.

Theorem 7.2. *The treewidth of control flow graphs is bounded by 2 for* **goto***-free Algol and Pascal, by 4 for Modula-2 and by 5 for* **goto***-free C programs.*

The importance of these control flow graphs and their treewidth is closely related to the register allocation problem. Observe that every variable that might appear in our program will be used within connected part of the the control flow graph. Each variable has to be assigned a register, and no two variables may share the same register. Hence we may define the intersection graph with variables as vertices. Two vertices form an edge, if they are in use at the same time, i.e., if the two parts of the control flow graphs corresponding to use of these two variables intersect. Then, the minimum number of registers needed for execution of the program is equal to the chromatic number of the intersection graph. Straightforwardly, any proper vertex coloring corresponds to a feasible register assignment to variables.

Observe that the fact that the control flow graph has bounded treewidth does *not* imply the same for the intersection graph. However, unlike general graph coloring problem that is hard to approximate (and decide for a fixed number of colors), under this assumption there were designed fast approximate algorithms for arbitrary number of registers [40] or linear-time exact (decision) algorithms for fixed number of registers [7].

References

[1] Aardal, K., Weismantel, R., and Wolsey, L. A. Non-standard approaches to integer programming. *Discrete Applied Mathematics 123*, 1–3 (2002), 5–74. Workshop on Discrete Optimization, DO'99 (Piscataway, NJ).

[2] Alon, N., Seymour, P. D., and Thomas, R. Planar separators. *SIAM J. Discrete Math. 7*, 2 (1994), 184–193.

[3] Archdeacon, D. A Kuratowski theorem for the projective plane. *Journal of Graph Theory 5* (1981), 243–246.

[4] Arnborg, S., Corneil, D. G., and Proskurowski, A. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods 8* (1987), 277–284.

[5] Arnborg, S., Proskurowski, A., and Corneil, D. Forbidden minors characterization of partial 3-trees. *Discrete Mathematics 80* (1990), 1–19.

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

[7] Bodlaender, H. L., Gustedt, J., and Telle, J. A. Linear-time register allocation for a fixed number of registers. In *ACM-SIAM Symposium on Discrete Algorithms, 9th SODA'98* (1998), pp. 574–583.

[8] Bouchitté, V., Mazoit, F., and Todinca, I. Chordal embeddings of planar graphs. *Discrete Math. 273*, 1-3 (2003), 85–102.

[9] Corneil, D. C., and Rotics, U. On the relationship between clique-width and treewidth (extended abstract). In *WG* (2001), A. Brandstädt and V. B. Le, Eds., vol. 2204 of *Lecture Notes in Computer Science*, Springer, pp. 78–90.

[10] Courcelle, B. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. 1990, pp. 193–242.

[11] Courcelle, B., Makowsky, J. A., and Rotics, U. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics 108*, 1–2 (2001), 23–52.

[12] Diestel, R. *Graph Theory*, graduate texts in mathematics ed., vol. 173. Springer-Verlag, May 1997. chapt. 12.

[13] Ebbinghaus, H.-D., and Flum, J. *Finite model theory*. Perspectives in mathematical logic. Springer, 1995.

[14] Fiala, J., Golovach, P. A., and Kratochvíl, J. Distance constrained labelings of graphs of bounded treewidth. In *ICALP* (2005), L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., vol. 3580 of *Lecture Notes in Computer Science*, Springer, pp. 360–372.

[15] Fiala, J., Golovach, P. A., and Kratochvíl, J. Parameterized complexity of coloring problems: Treewidth versus vertex cover. In *TAMC* (2009), J. Chen and S. B. Cooper, Eds., vol. 5532 of *Lecture Notes in Computer Science*, Springer, pp. 221–230.

[16] Halin, R. s-functions for graphs. *Journal of Geometry 8*, 1 (Mar 1976), 171–186.

[17] Kloks, T. *Treewidth. Computations and approximations*. No. 842 in Lecture Notes in Computer Science. Springer Verlag, 1994.

[18] Kuratowski, K. Sur le probleme des courbes gauches en topologie. *Fund. Math. 15* (1930), 271–283.

[19] Lampis, M. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica 64*, 1 (2012), 19–37.

[20] LaPaugh, A. S. Recontamination does not help to search a graph. *J. Assoc. Comput. Mach. 40*, 2 (1993), 224–245.

[21] Lapoire, D. Treewidth and duality for planar hypergraphs. 1996.

[22] Lenstra, Jr., H. W. Integer programming with a fixed number of variables. *Mathematics of Operations Research 8*, 4 (1983), 538–548.

[23] Lipton, R. J., and Tarjan, R. E. A separator theorem for planar graphs. *SIAM J. Appl. Math. 36* (1979), 177–189.

[24] Mohar, B. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math. 12*, 1 (1999), 6–26.

[25] Mohar, B., and Thomassen, C. *Graphs on surfaces*. Johns Hopkins University Press, Baltimore, 2001.

[26] Oum, S. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Ser. B 95*, 1 (2005), 79–100.

[27] Oum, S. Rank-width and well-quasi-ordering. *SIAM J. Discrete Math. 22* (03 2008), 666–682.

[28] Oum, S., and Seymour, P. D. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Ser. B 96*, 4 (2006), 514–528.

[29] Proskurowski, A. k-trees: representations and distances. *Discrete Mathematics 29* (1980), 785—794.

[30] Reed, B. A. Finding approximate separators and computing tree width quickly. In *ACM Symposium on Theory of Computing 24th STOC '92, Victoria, British Columbia, Canada* (1992), ACM, pp. 221–228.

[31] Reed, B. A. Rooted routing. manuscript, 2000.

[32] Reed, B. A. Algorithmic aspects of tree width. In *Recent advances in algorithms and combinatorics*, no. 11 in CMS Books in Mathematics. Springer Verlag, 2003, pp. 85–107.

[33] Robertson, N., and Seymour, P. D. Graph minors – III: Planar tree-width. *Journal of Combinatorial Theory, Series B 36*, 1 (1984), 49–64.

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

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

[36] Robertson, N., Seymour, P. D., and Thomas, R. Hadwiger's conjecture for K_6-free graphs. *Combinatorica 13*, 3 (1993), 279–361.

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

[38] Suchan, K., and Todinca, I. On powers of graphs of bounded NLC-width (clique-width). *Discrete Applied Mathematics 155*, 14 (2007), 1885–1893.

[39] Thomas, R. Tree decompositions of graphs. lecture notes, 1996.

[40] Thorup, M. Structured programs have small tree-width and good register allocation. In *WG* (1997), R. H. Möhring, Ed., vol. 1335 of *Lecture Notes in Computer Science*, Springer, pp. 318–332.

[41] Wanke, E. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics 54*, 2–3 (1994), 251–266.