Minors, Tree Decompositions and FPT Algorithms. MPRI

February 3, 2023

Contents

Introduction

These are notes in complement to a 9h lecture given in 2022-2023 in the MPRI (Master Parisien de Recherche en Informatique) about minors, treewidth and FPT algorithms on bounded treewidth classes of graphs. better

For a general textbook on graph theory, the book of Bondy and Murty ([4] is an excellent reading, but the theory of graphs minors and tree decomposition is not deeply discussed. On this particular topic, there exists a good a short introduction by Lovasz ([11]), but the main influence for this course is Chapter 12 of the book of Diestel ([7]), which is also a good source for exercises. For Fixed Parameter Tractable Algorithms, a classic book is ([8]).

Chapter 1

Introduction

1.1 Basic Definitions and Terminology

In this course, a graph is given by a set V, whose elements are called **vertices**, and a set E whose elements, called **edges** of the graphs, are distinct subsets of size 2 of V. According to the usual vocabulary, this means that our graph will always be simple and without loops. Unless specified, V will always be a finite set. For a graph G, $V(G)$ will always denote its set of vertices, $E(G)$ its set of edges. Very often we will write xy instead of $\{x, y\}$ for an edge of G.

A vertex v is a **neighbour** of a vertex u if $uv \in E(G)$. The **neighbourhood** of u, denoted $N(u)$ is the set of neighbours of u. Its **degree**, denoted $d(u)$ is the cardinality of its neighbourhood. The maximum degree of a graph is usually denoted Δ. A graph with no edges will be called a **stable set**, or **independent set**, and a graph will all possible edges between its vertices a **clique**, or **complete graph**. The complete graph on n vertices is usually denoted K_n.

The **path** P_k is a graph with $V(P_k) = \{x_1, x_2, \ldots, x_k\}$, with edges $E = \{x_i x_{i+1},\ 1 \leq i \leq k-1\}$. The vertices x_1 and x_k are called the **endpoints** of the path. If we add the edge $x_k x_1$ to P_k then the resulting graph is the **circuit** on k vertices, denoted C_k.

1.2 Three Algorithmic Problems

Consider the following problem of connectivity.
Problem : k **disjoint path problem**
Input : A graph G, an integer k and two subsets of vertices A and B of size k
Output : TRUE if there exists k vertex disjoint paths from A to B?

This problem is a very classical one, and Ford-Fulkerson Algorithm tells us that this is solvable in time $O((k|E(G)|)$ (classical Ford-Fulkerson Algorithm is for edge disjoint path in the directed case, but it is easy to reduce our case to this one). The maximum value k corresponds to a minimum vertex cut separating A and B and is a classical result of Menger.

Theorem 1.1 (*Menger,1927,[12]*)

> *Let x and y be distinct vertices of a graph G. Then the minimum number of vertices whose deletion separates x from y is equal to the maximum number of internally disjoint paths between*

x and y.

Proof. See [4]. □

Now consider the smilingly similar problem.

Problem : k-disjoint rooted path problem
Input : A graph G, an integer k, and two subsets of vertices $X = \{x_1, x_2, \ldots, x_k\}$ and $Y = \{y_1, y_2, \ldots, y_k\}$
Output : TRUE if there exists disjoint paths $P_1, P_2, \ldots, P_k$, such that P_i is a path from x_i to y_i.

This kind of problem in a more general form is known as commodity flow problem and has many applications. With k part of the input, this problem is NP-complete, even restricter to the class of planar graphs. Nevertheless, in the Graph Minor series of papers, Robertson and Seymour proved a polynomial algorithm for fixed k. This result is extremely difficult and relies and techniques and notions that will be illustrated in this course.

Theorem 1.2 (*Robertson-Seymour, [13]*)

The k-disjoint path problem can be solved in time $O\left((f(k).n^3\right)$

The result has been improved to quadratic time by Kawabayarashi, Kobashi and Reed ([]). Let us see an algorithmic consequence of this result related to topological minor detection.

Definition 1.3

*A graph H is **topological minor** of a graph G if there exists a injective mapping f from $V(H)$ to $V(G)$ such that for each edge uv of H, there exists in G a path P_{uv} connecting $f(u)$ and $f(v)$ in G with the property that all these path are internally disjoint.*

Example. Describe the graphs that do not contain the following graphs as topological minors : K_3, $K_{1,3}$, $K_{1,4}$.

A natural algorithmic problem is then the following.

Problem : Topological H-minor detection
Input : A graph G and a graph H.
Output : TRUE if H is a topological minor of G, FALSE otherwise.

The problem is NP-complete if H is part of the input, but if H but if H is fixed, then this problem was proven to be polynomial by Robertson and Seymour.

Theorem 1.4

Let H be a fixed graph. There exists a polynomial time algorithm to decide whether H is a topological minor of a given graph G.

Proof. Let $f : V(H) \to V(G)$ be an injection (note there are polynomially many such objects), we want to decide if there exists disjoint paths in G between the $f(v)$ corresponding to edges of H. To do that, we replace each vertex $f(v)$ by $d_H(v)$ copies of $f(v)$ (having the

same neighbours). Now, for $k = |E(H)|$, solving the k-Rooted Disjoint Path Problem for these sources clearly solves the desired question. □

The complexity of this algorithm is hence $O(f(k))\, n^k$, where k is the size of H, and n the size of G. It is therefore polynomial for every fixed k. In 2010, Gröhe, Kawabarayashi, Marx, and Wollan proved a stronger result, that this can be done in $O(f(k))\, n^3$. Such an algorithm is called Fixed Parameter Tractable (FPT) algorithm. We will discuss more about those in the last chapter of this course.

In particular, the previous theorem implies that any family of graphs that is defined with forbidding a FINITE family of graphs as topological minors is polynomially testable. One such family is very well known, it is the family of planar graphs, as was proven by Kuratowski in 1930.

Theorem 1.5 (*Kuratowski,1930*)

A graph G is planar if and only if it does not contain K_5 or $K_{3,3}$ as a topological minor.

Planar graph will play a central role in this course, and as we will see, this theorem will be characteristic of the kind of result we will be interested. The crucial fact here is not that planar graphs are defined by a certain list of forbidden topological minors, this is easy (why?), it is the finiteness of this list that is non trivial. The central result of Robertson and Seymour theory is that many different graph properties can be characterised by a **finite** list of forbidden substructures, and hence get polynomial time recognition algorithm. Of course, one does not need the difficult of Robertson and Seymour to prove that planar graphs are polynomially recognisable, there even exists linear time algorithm to do that. Nevertheless, we will see later that there are instances of such recognition problem for which the only proof of polynomiality was obtained through their results.

1.3 Minors

We define three operations on a graph G (at the end of each line the notation for the resulting graph).

1. **Remove a vertex** v (and all its incident edges) : $G \setminus v$
2. **Remove an edge** e (but not its end vertices) : $G \setminus e$
3. **Contract an edge** $e = xy$, which means remove x and y, add a new vertex z whose neighbourhood is the union of the neighbourhoods of x and y (without putting any loop on z) : G/e.

A contraction G/e is **topological** if one of the endpoints of e has degree 2. Its inverse is the subdivision operation which consists in removing an edge xy, adding a new vertex z, and adding the edges xz and zy.

Definition 1.6

Let G and H be two graphs.

- *H is an **induced subgraph** of G if H is obtained from G by the repeated use of rule 1.*
- *H is a **subgraph** of G if H is obtained from G by the repeated use of rule 1 and 2.*

- *H is a **spanning subgraph** of G if H is obtained from G by the repeated use of rule 2.*
- *H is a **minor** of G if H is obtained from G by the repeated use of rule 1,2 and 3.*
- *H is a **topological minor** of G is H is a minor of G and every contraction used was topological.*

Recall that the largest integer k such that G has a complete graph (resp. independent set) on k vertices as an (induced) subgraph, is called the **clique number** (resp. **independence number**) of G, denoted $\omega(G)$ (resp. $\alpha(G)$). Due to a classical result of Karp ([10]), deciding if a graph has $\omega(G) \geq k$ or not (similarly for $\alpha(G)$) is an NP-hard problem.

The following lemma gives an alternate definition of minor that is often useful.

Lemma 1.7

Let G and H be two graphs, and denote $V(H) = \{v_1, \ldots, v_p\}$. Then H is a minor of G if and only if there exists p connected and disjoint subgraphs $G_1, \ldots, G_p$ of G such that for every edge v_iv_j of H, there exists an edge between G_i and G_j.

Let us mention here that high density implies the existence of a large minor; there exists theorems with better bounds, but we are only interested here in the fact that such a bounds exists.

Theorem 1.8

Every graph with average degree at least 2^{r-2} contains K_r as a minor.,

Proof. By induction on r. Let G be a graph of average degree at least 2^{r-2}. Therefore $|E(G)|/|V(G)| \geq 2^{r-3}$. Let H be minimal amongst all minors of G such that $|E(H)|/|V(H)| \geq 2^{r-3}$. It implies that when one contracts an edge in H, one must loose at least 2^{r-3} edges (otherwise the inequality would still be satisfied, and H would not be minor minimal). Hence, for any xy edge of H, x and y have at least 2^{r-3} common neighbours. In other words, if x is a vertex in H, then the minimum degree in its neighbourhood is at least 2^{r-3}, so by induction it contains a K_{r-1} minor, which yields with x the desired K_r minor. □

Let us discuss now the difference between minors and topological minors. Topological minors are special kind of minors but of course the converse is not true : a graph G can contain H as a minor, but not as a topological minor. (Exercise **??**).

When H is of small maximum degree, this is nevertheless true.

Theorem 1.9

Let H be a graph with maximum degree at most 3. Then a graph G has an H-minor if and only if it contains an H-subdivision.

Proof. Let H be a graph with vertex set $V(H) = \{v_1, \ldots, v_p\}$ and assume it is a minor of G. We use Lemma 1.7 : there exists p connected and disjoint subgraphs $G_1, \ldots, G_p$ of G such that for every edge v_iv_j of H, there exists an edge between G_i and G_j.

Any topological minor of the graph induced by the G_i will be topological minor of G so we apply topological minor operations. First between the G_i we delete all edges but one between each pair when it is needed, that is when there was one in H between the

corresponding vertices. Then we can delete edges inside each G_i keeping it connected and therefore assume that every subgraph G_i is a tree. Also, every leaf of such a tree must be a vertex incident to an edge going to another G_i, otherwise it is not needed. Finally we can contract edges to get rid of degree 2 vertices. It is easy to see that what we get is then a single vertex in each G_i and therefore the graph H as a topological minor of G.

□

In the general case, using the same argument we are able to still prove a result that reduces minor detection to topological minor detection.

Theorem 1.10

For every graph H, there is a finite family $\mathcal{H}$ of graphs with the property that G contains H as a minor if and only if it contains some graph in $\mathcal{H}$ as a topological minor.

Proof. We start the proof exactly as in the previous result, and by again choosing minimal G_i, we now get for each G_i a tree with at most $|H|$ leaves and no vertex of degree 2. There finitely many such trees (why?). So by replacing the vertices of H by these trees in all possible ways, we obtain a finite collection of graphs $\mathcal{H}$ with the desired properties.

□

This result combined with Theorem 1.4 now clearly implies the following theorem due to Robertson and Seymour (Graph Minors XIII [15])

Theorem 1.11 (*Robertson and Seymour,1995*)

Let H be a fixed graph. There exists a polynomial time algorithm to decide whether H is a minor of a given graph G.

Of course this extends to the recognition of any family of graphs that is defined by a finite list of forbidden minors. The question is then to understand what are the families of graphs of this type. A trivial fact is that such families are **closed under minors** (every minor of a graph in the family is in the family). The celebrated Robertson and Seymour Theorem, solving a conjecture of Wagner from 1937 says that this is sufficient.

Theorem 1.12 (*Robertson and Seymour*)

A minor closed class of graph is defined by a finite list of forbidden minors

The proof of this conjecture is due to Robertson and Seymour, and is the main result of a series of more than 20 papers called the Graph Minors series, written in the 1980's. The conjecture of Wagner is eventually proven in Graph Minors XX ([16]), published in 2004.

One of the most important special case of the conjecture is the generalisation of Kuratowski's Theorem for any fixed surface. It was first proven by Archdeacon and Huneke for non-orientable surfaces ([1]), and for the general case by Robertson and Seymour in Graph Minor VIII [14].

Theorem 1.13

For every closed compact surface, the set of graphs that are embeddable in this surface has a finite set of bounds.

Another example is the question of linklessly embeddable graphs (graphs that can be embedded in three-dimensional spaces, so that two cycles can always be pulled apart, that is they don't entwine like links in a chain). Before Robertson and Seymour result, it was an open problem to know if these are defined by a finite list of forbidden minors, or recognisable in polynomial time.

Combining the theorems above, one gets the important fact that any closed-minor property is polynomially testable.

Chapter 2

Minor Closed Classes and Well Quasi Orders

2.1 Wagner's Conjecture - Well Quasi Orders

Recall that a class of graphs is minor-closed if every minor of a graph of the class is also in the class. In Chapter 1, we discussed Robertson and Seymour proof of a conjecture of Wagner :

Conjecture 2.1 (*Wagner*)

For every minor closed class C, there exists a finite set of graphs $\mathcal{F}_C$ (often called obstructions) such that a graph belongs to C if and only if it does not contain (as a minor) any graph in $\mathcal{F}_C$.

Let us discuss here what can these obstructions be. For a given minor closed class C, a graph H is said to be a **bound** if G is not in C but every strict minor of G is. Note that if H is a bound, since it is not in C it must contain any obstruction so it must be itself an obstruction. The following easy proposition tells us that the set of bounds is in fact a sufficient set of obstructions.

Proposition 2.2

Let C be a minor closed class, and $\mathcal{B}$ be its (possibly infinite) set of bounds. Then $G \in C$ if and only if G does not contain any graph of $\mathcal{X}$ as a minor.

Proof. If H not in the class then either it is minimal or it contains H' not in the class. We can repeat the argument, and since there exists no infinitely decreasing sequence of graphs, every graph not in the class admits one of the bound as a minor. □

(This uses the fact that there exists no infinite decreasing sequence of graphs for the minor order - such partial orders are called **well founded**.)

As said before, it implies that testing if a certain graph G belongs to C is exactly testing if G contains one of the minor-minimal graphs with respect to C. Here is a table describing the set of minor-minimal graphs for certain classes.

Graph Class	Minor minimal graphs
Forests	triangle
Union of Paths	triangle, claw
Planar	K_5 , $K_{3,3}$
Toric	$\geq$ 16629 (but finite)

So another way of stating Wagner conjecture would be to say : **Every minor closed class of graphs has a finite set of bounds**. Note that by definition one bound cannot be the minor of another. Using the terminology of partially ordered sets, they form an **antichain** : a set of pairwise not comparable elements. So a way to prove Wagner conjecture would be to prove that there exists no infinite antichain for the minor relation on graphs. In fact this equivalent as we will show now.

Definition 2.3

*A partial order $\preccurlyeq$ defined on a set X is a **well quasi order** (WQO) if there is no infinite decreasing sequence and no any infinite antichain.*

A infinite sequence that is either decreasing or an antichain will always be called a **bad sequence**. A wqo is hence defined as a partial order with no bad sequences. Note that in the case of graphs, there cannot be an infinite decreasing sequence, so the only possible bad sequence would be an infinite antichain.

Proposition 2.4

Wagner's conjecture 2.1 is equivalent to say that the class of all graphs with the minor relation is a wqo.

Proof. Assume that the minor relation is a wqo. consider a class C that is minor closed. Let F_C be the class of graphs minimally (for the minor relation) not in C. Then F_C is an antichain, so it is finite, and it is easy to see that G is in C if and only if G does not contain any graph in F_C as a minor Now assume that Wagner's conjecture is true. Assume there exists a bad sequence of graphs G_n. Let C be the class that do not contain any of the G_n as a minor. It is minor closed, hence there exists a finite list (H_i) such that G is in C iff it does not contain any of the H_i as a minor. So every G_i must contain one of these graphs as a minor. By pigeonhole principle, there exists G_i and G_j that contain the same H_k. But conversely, H_k is not in C so by definition it must contain one of the G_n as a minor. by transitivity, this contradicts the fact that the G_n form an antichain.

□

In the next section we will try to understand some of the ideas behind the proof of Wagner's conjecture by proving similar but (much) easier results, the main of which being a theorem due to Kruskal saying that trees are well quasi ordered for the minor relation.

2.2 Words, Paths and trees

Before stating the first result of this section about wqo, we prove the following proposition, that will be of use all through the section.

Proposition 2.5

Let $(X, \preccurlyeq)$ be a partially ordered set and $(x_i)_{i\in\mathbb{N}}$ be any sequence. Then this sequence has a infinite subsequence that is either increasing, or decreasing or an antichain.

Proof. Let (x_i) be any sequence. Start with x_1, and consider

- $A_1 = \{j,\ j > 1 \text{ and } x_1 \preccurlyeq x_j\}$

- $B_1 = \{j,\ j > 1 \text{ and } x_1 \preccurlyeq x_i\}$
- $C_1 = \{j,\ j > 1 \text{ and } x_1 \text{ and } x_j \text{ are incomparable}\}$

If A_1 is infinite we say that x_1 is of type A and delete all elements that are not in A_1. If not, but B_1 is infinite, say that x_1 is of type B and delete all elements that are not in B_1. Finally in the last case, say that x_1 is of type C and delete all vertices not in C_1.

Up to extracting a subsequence and renaming, we can assume no elements were deleted, so that all x_i with $i \geq 2$ were in A_1, or B_1, or C_1. We do this sequentially of x_2, then x_3, ... At each step, we define A_i, B_i, C_i as

- $A_i = \{j,\ j > i \text{ and } x_i \preccurlyeq x_j\}$
- $B_i = \{j,\ j > i \text{ and } x_j \preccurlyeq x_i\}$
- $C_i = \{j,\ j > i \text{ and } x_i \text{ and } x_j \text{ are incomparable}\}$

and at each step we define the type of x_i to be one of A, B, C depending on which is infinite. Then we extract by keeping only the elements in the infinite set.

Eventually we have a type for each element of the sequence (which is in fact an subsequence of the original sequence). Now there must be a type with infinitely number of elements and to each type clearly corresponds one of the three possible type of infinite subsequence. □

From this we deduce the following which gives equivalent conditions for being a wqo.

Corollary 2.6

Let $(X, \preccurlyeq)$ be a partially ordered set. The three assertions are equivalent

1. *$(X, \preccurlyeq)$ is a wqo*
2. *from every sequence $(x_i)_{i\in\mathbb{N}}$ one can extract an infinite increasing subsequence.*
3. *from every sequence $(x_i)_{i\in\mathbb{N}}$ one can extract $i < j$ such that $x_i \preccurlyeq x_j$.*

This will be useful : in order to prove that a given partial order is a wqo, we will only prove the third statement, but when we use the fact that an order is a wqo (for example in a proof by induction), we can use the second statement which is (in appearance) much stronger.

Let us illustrate this by proving that the set of words on a finite alphabet is wqo for the subword relation.

Proposition 2.7

Assume X is a finite set, and define X^* as the set of finite sequence of elements of X. Define a partial order $\preccurlyeq$ on X^* by $u \preccurlyeq v$ if u is a subword of v (u can be obtained from v by deletion of letters). Then $(X^*, \preccurlyeq)$ is a wqo.

Proof. We do a proof by induction on the size of X. Assume the opposite, and let $w_1, w_2, \ldots$ be a bad sequence. It implies in particular that $w_1 \not\preccurlyeq w_i$ for every $i \geq 2$. This will help us to gain some structure on the w_i. Let $w_1 = x_1 x_2 \ldots x_k$ and define the function $f : X^{(\mathbb{N})} \to \{0, 1, \ldots, k\}$ by $f(w) = \max\{i,\ x_1 x_2 \ldots x_i \preccurlyeq w\}$ (by convention $f(w) = 0$ if no

such i exists). There must be an integer $0 \leq j \leq k-1$ such that $\{i,\ f(w_i) = j\}$ is infinite. To simplify notations and avoid considering a subsequence, we assume in fact that $f(w_i) = j$ for all i.

Look now at a word w with $f(w) = j$; It means it can be decomposed in $w = u_1 x_1 u_2 x_2 \dots u_j x_j u_{j+1}$, for all i, and u_i are words such that $x_i \not\preccurlyeq u_i$ for all i between 1 and $j+1$. Each u_i thus belong to the set of words on the alphabet $X_i = \{x \in X,\ x_i \not\preccurlyeq x\}$. By induction these sets of words are wqo for $\preccurlyeq$, then by extracting an infinite increasing subsequence in the u_1^i, then extracting from this subsequence another infinite increasing subsequence for u_2^i, etc.. one can get an infinite increasing subsequence in the w_i, which contradicts the hypothesis.

□

One can remark that words with the subword relation correspond to paths (as graphs) with labels on the vertices. A word is a subword of another, if the first path admits a subdivision that is a subpath of the second, with the same labels. In other words, we have proven that labelled paths are wqo for the topological minor with labels order. We will soon prove this for trees.

Higman Theorem replaces the word "finite" by WQO :

Proposition 2.8 (*Higman, 1952, [9]*)

Let $(X, \preccurlyeq)$ be a wqo. Extend this partial order to X^* by $x_1 x_2 \dots x_k \preccurlyeq y_1 y_2 \dots y_l$ if there exists an increasing injection $f : \{1, \dots, k\} \to \{1, \dots, l\}$ such that for every i, $x_i \preccurlyeq y_{f(i)}$. Then $(X^*, \preccurlyeq)$ is also a wqo.

Note that it implies the previous one : if X is finite, it suffices to consider on X the partial order where no two elements are comparable - it clearly defines a wqo.

Proof. Assume by contradiction the existence of bad sequences, and choose a minimal bad sequence (MBS) in the following sense : if $w_1, w_2, \dots, w_i$ are defined, w_{i+1} is chosen as minimal (with respect to its length), amongst the words w for which there exists a bad sequence starting with $w_1, w_2, \dots, w_i, w$. (such a word always exists as there is no infinite decreasing sequence here).

Let $w_i = x_i w_i'$. Assume the (w_i') contains a bad subsequence $w'_{\phi(i)}$, where ϕ is in increasing function. Consider the sequence $w_1, w_2, \dots, w_{\phi(1)-1}, w'_{\phi(1)}, w'_{\phi(2)}, w'_{\phi(3)}, \dots$. It is easy to see that it is also a bad sequence, contradicting the minimality of (w_i). Therefore (w_i') contains no bad subsequence. It implies that it contains a increasing subsequence, but now since X is a wqo, it exists two indices i and j from this subsequence such that $x_i \preccurlyeq x_j$, which yields $w_i \preccurlyeq w_j$, contradicting the hypothesis.

□

In fact we will not use exactly the previous proposition, but the following that deals with finite subsets. The proof is almost identical.

Proposition 2.9 (*Higman, 1952, [9]*)

Let $(X, \preccurlyeq)$ be a wqo. Extend this partial order to $X^{(<\infty)}$ (finite subsets of X) by $A \preccurlyeq B$ if there exists an injection $f : A \to B$ such that for every $a \in A$, $a \preccurlyeq f(a)$. Then $(X^{(<\infty)}, \preccurlyeq)$ is also a wqo.

We are now ready to move on to the next step, trees. A famous result from Kruskal is the following.

Theorem 2.10

Finite trees are well quasi ordered for the topological minor relation

Notice that it is stronger than the same statement with the standard minor relation. Here we are going to prove an even stronger result, because we are going to consider topological minor on labelled rooted trees : we consider finite trees whose nodes are labelled by elements from a wqo $(X, \leqslant)$. As in the case of words for Higman's Lemma, we extend this partial order on W to a partial order on those labelled rooted trees. Let T and T' two such trees.

We say that $T \leqslant T'$ if there exists a subdivision T'' of T with an isomorphism ϕ between T'' to a subgraph of T' such that

- ϕ preserves the tree order : if x is a ancestor of y in T, then $\phi(x)$ is an ancestor of $\phi(y)$ in T'.
- ϕ preserves the labels : for every vertex x of T, $label(x) \leqslant label(\phi(x))$ (we don't consider for this the vertices of $T'' \setminus T$ obtained by subdivisions, they don't have any label anyway)

If X has cardinality one, this is exactly the topological minor relation for rooted trees. Note also that this partial order restricted to paths is exactly the partial order defined on words in Higman's Theorem 2.8. The proof given below is due to Nash-Williams.

Theorem 2.11 (*Kruskal*)

Consider finite labelled trees and the partial order $\leqslant$ as in the above paragraph. Then it defines a well quasi order.

Proof. Assume there exists a bad sequence of trees for this order, and consider a sequence T_n of trees that is a minimal bad sequence, which is defined the following way : if $T_0, T_1, \ldots, T_i$ are defined, T_{i+1} is chosen as a minimal tree (with respect to its size) such that there exists an infinite bad sequence starting with $T_0, \ldots, T_{i+1}$.

Denote by r_i the root of T_i, and $\mathcal{U}_i$ the family of trees obtained by removing r_i from T_i. Let $\mathcal{U}$ be the union of all $\mathcal{U}_i$.

We first prove that $\mathcal{U}$ is well quasi ordered. Assume for contradiction that Z contains a bad sequence. Up to taking a subsequence, we can assume that there exists a bad sequence $U_{\phi(i)}$ such that for all i, $U_{\phi(i)} \in \mathcal{U}_{\phi(i)}$. Consider the sequence $T_0, T_1, \ldots, T_{\phi(0)-1}, U_{\phi(0)}, U_{\phi(1)}, U_{\phi(2)} \ldots$. It is easy to see that it is a bad sequence, which contradicts the minimality of the T_i.

Therefore, this set $\mathcal{U}$ is well quasi ordered, and by Theorem 2.9, the set of finite subsets of $\mathcal{U}$ is also wqo. Therefore the sequence $(\mathcal{U}_i)_{i\in\mathbb{N}}$ admits an infintie increasing subsequence $(\mathcal{U}_{\psi(i)})_{i\in\mathbb{N}}$ (remember that the order on subsets is the one defined in Theorem 2.9). Now as in Higman's Theorem, we look at the sequence of labels of the roots $r_{\psi(i)}$ Since the set of labels is wqo, there exists $i < j$ such that $l(a_{\psi(i)}) \leqslant l(a_{\psi(i)})$. Together with $\mathcal{U}_{\psi(i)} \leqslant \mathcal{U}_{\psi(j)}$, this gives $T_i \leqslant T_j$, and our contradiction.

□

Chapter 3

Tree Width

3.1 Definitions

Definition 3.1

- *Let G be a graph. A tree decomposition of G is a pair (T, W), where T is a tree and $W = (W_t)_{t \in V(T)}$ a collection of subsets of $V(G)$ satisfying :*
  - *For every $u \in V(G)$, $T_u = \{t \in V(T)\ ,\ u \in W_t\}$ induces a connected subgraph of T.*
  - *For every edge $uv \in E(G)$, $T_u \cap T_v \neq \emptyset$.*
- *The **width** of a tree decomposition is $\max_{t \in V(T)}(|W_t| - 1)$*
- *The **tree width** of a graph G, denoted $\mathbf{tw(G)}$, is the minimum width of a tree decomposition of G*

Equivalently, a tree decomposition of G is a tree T along with a collection of subtrees T_v, one for each vertex of G, with the condition that T_u and T_v intersect if uv is an edge of G. Note that is not a equivalence, it is possible that $T_u \cap T_v \neq \emptyset$ even if $uv \notin E(G)$ (this will be the case for chordal graphs).

Here is a key lemma regarding subtree intersection; by analogy with Helly's Theorem on convex subsets of $\mathbb{R}^d$, this property is often called Helly property of subtrees of a tree.

Lemma 3.2

Let $\mathcal{F}$ be a collection of pairwise intersecting subtrees of a given tree T. Then $\cap_{T \in \mathcal{F}} T \neq \emptyset$.

Proof. If not, for each vertex t of the tree, there is a subtree in $\mathcal{F}$ that does not intersect this vertex, and therefore is contained in one of the components of $T \setminus t$. One edge incident to t corresponds to this component, orient this edge out from t. One gets this way an orientation of some edges of T such that each vertex has exactly one outgoing edge. Since there are less edges than vertices in a tree, there must be an edge oriented both ways, which results in two non intersecting subtrees in $\mathcal{F}$. Contradiction. □

Corollary 3.3

Let G be a graph and K be complete subgraph of G. In any tree decomposition (T, W) of G, there exists a vertex t of T such that $K \subset W_t$. In particular, $\mathrm{tw}(G) \geq \omega(G) - 1$

Or course tree decompositions, even optimal ones, are not unique. Let us try to define a way to somehow minimise an optimal decomposition. Assume there are two adjacent vertices s and t of T such that $W_s \subset W_t$. Then we can contract the edge st of T to a new vertex r, and define $W_r = W_t$. It is trivial to check that this defines a valid tree decomposition with width no larger than the original one. By repeating this operation we obtain the following proposition.

Proposition 3.4

For every graph G, there exists a tree decomposition of width $\mathrm{tw}(G)$ *such that for every edge $st \in E(T)$, $W_s \not\subset W_t$ and $W_t \not\subset W_s$. In particular, for every leaf $f \in V(T)$, there exists a vertex $u \in V(G)$ such that $T_u = \{f\}$.*

Theorem 3.5

In every graph G, there exists a vertex of degree at most $\mathrm{tw}(G)$.

Corollary 3.6

$\chi(G) \leq \mathrm{tw}(G) + 1$

3.2 Treewidth and Minors

Proposition 3.7

If H is a minor of G, then $\mathrm{tw}(H) \leq \mathrm{tw}(G)$

Proof. It is enough to prove that the three operations of vertex deletion, edge deletion and edge contraction cannot increase treewidth. So starting from an optimal tree decomposition of G, we find one for the new graph without increasing the size of the bags.

- for $G \setminus e$, do nothing
- for $G \setminus v$, juste remove T_v.
- for G/e, where $e = uv$: the new vertex is called w. Delete T_u and T_v, set $T_w = T_u \cup T_v$

□

Corollary 3.8

The class of graphs of treewidth at most k is closed under taking minors.

Therefore, using the theorem of Robertson and Seymour, we know that it is defined by a finite number of excluded minors. In fact, this result is not a consequence of their theorem, but one of its steps, as this result can be proven directly using the ideas of the proof of Kruskal Theorem 2.11.

Theorem 3.9

The class of graphs of treewidth at most k is well quasi order for the minor relation

Proof. Admitted. Uses a variant of Kruskal Theorem. □

Corollary 3.10

The class of graphs of treewidth at most k has a finite number of bounds.

Proof. If G is a bound for the class of tw $\leq k$, then $\text{tw}(G) = k + 1$ (why?). So the set of bounds is an antichain contained in the class of graphs of treewidth at most $k + 1$. Hence it is finite by Theorem 3.9. □

Let us try to describe the bounds for small values of k.

Theorem 3.11

- $\text{tw}(G) \leq 1 \Leftrightarrow$ *G is a forest* $\Leftrightarrow$ *G does not contain K_3 as a minor*
- $\text{tw}(G) \leq 2 \Leftrightarrow$ *G does not contain K_4 as a minor*

Proof. The first item is trivial, and for the second, one direction is easy : since $\text{tw}(K_4) = 3$, graphs of tree width at most 2 are indeed K_4-minor free.

So let us prove that if G is K_4 minor free, then it has treewidth at most 2. First we can assume that G is not 3 connected. Indeed, assume by contradiction that G is 3-connected. Let C be a minimum cycle in G. If $G = C$, then $\text{tw}(G) = 2$. If not, there exists $x \in G \setminus C$, and by Menger's Theorem, there exists three disjoint paths from x to C. Using these paths and the cycle, one gets a K_4 minor.

Therefore G has a separator of size at most 2. Assume G has a separator $\{a, b\}$ of size 2 where $ab \notin E$. It is easy to see that one can add the edge ab without creating a K_4 minor. Hence we can add the edge $e = ab$ and the graph $G + e$ still has np K_4-minor, and $G + e$ has a clique separator of size 2. So in all cases we can assume that G has a separator S which is a clique (of size 1 or 2), and we can glue he decompositions of the components of $G \setminus e$ to conclude by induction that G has treewidth at most 2. □

This proofs shows the role of separators with treewidth. The operation of gluing graphs along cliques is called clique sum and will be described in the next section.

One could hope for a general result $\text{tw}(G) = k$ iff G does not have a K_{k+2} minor. Unfortunately, this fails for $k = 3$. There exists graph with no K_5 minor (and as we will see soon, planar graphs) with arbitrarily high treewidth.

Due to the following theorem, there are in fact four bounds for tree width at most 3. O is the octahedron, W_8 is the cycle on 8 vertices where all edges linking diametrically opposite nodes are added, and $C_5 \times K_2$ is the pentagonal prism (two C_5 joined by a perfect matching).

Theorem 3.12

$\text{tw}(G) \leq 3 \Leftrightarrow$ *G does not contain one of the four following graphs as as a minor : K_5,W_8, O and $C_5 \times K_2$.*

3.3 Digression : Hadwiger Conjecture

We have already proven these two sets of inequalities

$$\begin{aligned} \omega(G) \leq \quad \chi(G) \quad &\leq \mathrm{tw}(G) + 1 \\ \omega(G) \leq \quad \omega_m(G) \quad &\leq \mathrm{tw}(G) + 1 \end{aligned}$$

where $\omega_m(G)$ denotes the largest integer k such that G has a K_k minor. Now, note that Hadwiger Conjecture **??**, can be formulated exactly as $\chi(G) \leq \omega_m(G)$. Let us examine rapidly the easy cases.

It is trivial that $\omega_m(G) \leq 2 \Leftrightarrow G$ is a forest $\Rightarrow \chi(G) \leq 2$.

As we have seen before $\omega_m(G) \leq 3 \Leftrightarrow \mathrm{tw}(G) \leq 2 \Rightarrow \chi(G) \leq 3$ by the above inequalities.

Note that the next case, $\omega_m(G) \leq 4 \Rightarrow \chi(G) \leq 4$ implies the Four Colour Theorem since planar graphs are K_5-minor free. In fact it is equivalent (and hence true), thanks to a structural characterisation of graphs with no K_5 minor due to Wagner.

3.4 Separators

The purpose here is to prove that indeed tree decomposition 'decompose' the graph. Let us state first an easy but fundamental result.

Proposition 3.13

Let (T, W) be a tree decomposition of G and t_1t_2 be an edge of T and denote by S the set of vertices $W_{t_1} \cap W_{t_2}$. For $i = 1, 2$, define T_i as the connected component of $T \setminus t_1t_2$ containing t_i, and G_i the subgraph of G induced by $\cup_{t \in T_i}(W_t \setminus S)$. Then there are no edges between G_1 and G_2.

This can be stated also the following way

Theorem 3.14

If H is a connected subgraph of G, then for any tree decomposition (T, W) of G, $\cup_{v \in V(H)} T_v$ induces a subtree of T.

Conversely, we have the following result about cutsets that induce complete graphs.

Proposition 3.15

Let G be a graph with a clique cutset S and let $(X_i)_{i \in I}$ be the connected components of $G \setminus S$. Define H_i to be the graph induced by $X_i \cup S$. Then $\mathrm{tw}(G) = max_{i \in I}(\mathrm{tw}(H_i))$.

Definition 3.16

*Let G_1 and G_2 be two graphs and K_1 a clique of G_1, K_2 a clique of G_2 with $|K_1| = |K_2|$. If G is a graph obtained by identifying vertices of K_1 and K_2, and then removing some edges of this clique, then G is a **clique sum** of G_1 and G_2.*

Again Proposition 3.15 can be restated in terms of clique sums.

Proposition 3.17

If G is a clique sum of G_1 and G_2, then $\mathrm{tw}(G) \leq \max(\mathrm{tw}(G_1), \mathrm{tw}(G_2))$.

We have seen that K_4-minor free graphs are exactly graphs with treewidth at most 2. Hence they have separators of size at most 1 and 2. In fact they are exactly the graphs obtained from K_2 by operations called series and parallel (see figure below) and are often called for these reasons **series-parallel** graphs.

Now we are going to give a proof that planar free graphs are not bounded for treewidth, by proving that grids can have arbitrarily high treewidth. To do so we begin with the following proposition, which gives a lower bound on the treewidth in terms of minimum good separator. This proposition will also be very useful to design algorithms.

Proposition 3.18

Let G be a graph of tree width k. Then there exists a subset $X \subset V(G)$ such that

- *X is a cutset of G*
- *X has size at most $k+1$*
- *no connected component of $G \setminus X$ has size larger than $|V(G)|/2$*

Proof. First we can assume the graph is connected, for we can always apply this result on the biggest connected component. Now let (T, W) be an optimal tree decomposition and t_1t_2 be an edge of T. Let us use the notations of Proposition 3.13, and orient t_1t_2 towards t_i if G_i contains more than $n/2$ vertices. The edge cannot be oriented in both directions otherwise the would be no vertices in $W_{t_1} \cap W_{t_2}$ and G would not be connected. Therefore there exists a vertex $t \in T$ such that no edge incident to t is oriented out from t. The bag W_t is the desired set of vertices. □

Now we are able to prove that the $2k \times 2k$-grid has treewidth at least k (in fact we will prove in section 3.5, that the treewidth is $2k$).

Corollary 3.19

The treewidth of the grid $G_{2k,2k}$ is at least k.

Proof. Assume the contrary, use Proposition 3.18 to get a good cut of size at most k. There are at least k rows and 1 column that this set misses. Since these rows and columns form a connected set of vertices, they are all in the same component which has therefore size at least $2k^2 \geq |G|/2$. Contradiction. □

Corollary 3.20

The class of planar graph has unbounded treewidth.

3.5 Duality - Cops and Robbers

As we have seen at the beginning of this chapter, if G is chordal, then its treewidth is precisely equal to $\omega(G) - 1$. Therefore, to certify that some decomposition is optimal, it is sufficient in that case to show in the graph some clique of the appropriate size. For a general graph G, one can use the largest order of a clique minor, or the largest k such that a k, k grid is a minor, as a lower bound on $\text{tw}(G)$. However, these bounds can be far from the exact value. For example grids have arbitrarily large treewidth, but no clique minor of order more than 4. The purpose of this section is to provide a dual notion for treewidth, that is something that certifies an exact lower bound.

Note that in the proof of Proposition 3.18, the crucial property is that the set of connected subgraphs of size at least $n/2$ satisfy the following : either they intersect, or there is an edge from one to the other (the whole graph is connected). For any collection of such connected subsets that satisfy this, there exists one bag of the decomposition that intersects all of them. This is precisely the notion behind the dual of treewidth.

Definition 3.21

- *We say that two connected subgraphs of G **touch** if they have non empty intersection or if they are joined by an edge.*
- *A **bramble** of G is a collection $\mathcal{B}$ of connected subgraphs that are pairwise touching.*
- *A **transversal** of a bramble $\mathcal{B}$ is a set of vertices of G that has non empty intersection with each element of $\mathcal{B}$.*
- *The **order** of a bramble $\mathcal{B}$ is the minimum size of a transversal of $\mathcal{B}$.*
- *The **bramble number** of G, denoted $\text{bn}(G)$, is the maximum order of a bramble of G.*

This notion was introduced by Seymour and Thomas and is the dual notion of treewidth, as the following theorem proves.

Theorem 3.22 (*Seymour and Thomas, 1993 []*)

For every graph G, $\text{bn}(G) = \text{tw}(G) + 1$

This theorem is a sort of minmax theorem (in fact maxmin=minmax).

As we have explained before the definitions, to prove the lower bound one just needs to mimic the proof of Proposition 3.18, which is concerned by the bramble formed by connected subgraphs of size at least $|V(G)|/2$. In a few paragraphs, we will give a different proof of this using a 2 player game one the graph. The upper bound is harder to prove. In the next chapter we will prove an approximate theorem, namely that $\text{tw}(G) \leq 4\text{bn}(G)$.

Let us illustrate the lower bound with the grid $G_{n,n}$. We already know that its tree width is at most n. So with the previous theorem, we only need to exhibit some bramble of order $n+1$. Denote by x_{ij} the vertex on i-th row and j-th column. Define a bramble whose sets are $\{A, B\} \cup \{C_{ij}, 1 \leq i, j < n\}$, where :

- $A = \{x_{i,1},\ 1 \leq i \leq n\}$, the last row,
- $B = \{x_{1,j},\ 1 \leq j < n\}$ the last column minus its last element,
- $C_{ij} = \{x_{kj},\ \ 1 \leq k < n\} \cup \{x_{ik},\ \ 1 \leq k < n,\}$.

It is easy to check that this constitutes a bramble of order $n + 1$.

Now we want to do an alternate proof of the lower bound by introducing an third invariant $\mathrm{cn}(G)$, and prove the two inequalities $\mathrm{tw}(G) + 1 \geq \mathrm{cn}(G)$ and $\mathrm{cn}(G) \geq \mathrm{bn}(G)$. This will come from the fact that $\mathrm{cn}(G)$ will be defined through a two player game, where tree decomposition corresponds exactly to strategies for one player, and brambles correspond exactly to strategies for the second player.

Let us define what a cops and robber game on a graph is. In that kind of two player game, one is controlling several cops, the other one is controlling the robber, and the goal of the first player is to capture the robber. There are many variants of this game (see [] for a state of the art), depending on where the cops/robber can move, on how they move, at which speed, or on the type of game (turn by turn or simultaneous). Here we are describing a variant that was introduced by Seymour and Thomas in [17].

In this variant, cops are standing on vertices of the graph, and at each turn a fraction of them can move by helicopter and land on any vertex of the graph. The robber sees the helicopter approaching and can instantly move at infinite speed to any other vertex along a path of a graph. The only constraint is that he is not permitted to run through a vertex occupied by some cop. The cops win if they capture the robber, so they win if at some point they occupy all vertices adjacent to the position of the robber, and an extra cop lands by helicopter on the robber.

For a graph G, the **cop number** of G, denoted $\mathrm{cn}(G)$ is the smallest number of cops to ensure the capture of the robber.

Proposition 3.23

$$\mathrm{cn}(G) \leq \mathrm{tw}(G) + 1$$

Proof. Let (T, W) be a tree decomposition of G with width $\mathrm{tw}(G)$. Put every cop one the vertices of some bag W_t. The robber, if it escapes has to be in some vertex appearing only in the bags of some component of $T \setminus t$. Let t' the neighbour of t in T in the direction of this component. Thanks to Proposition 3.13 $W_t \cap W_{t'}$ separates the component containing he robber form the rest of the graph. At the next move, the player controlling moves cops in $W_t \setminus W_{t'}$ to occupy all of $W_{t'}$. He now keeps on applying this strategy until it reaches some leaf of the tree and the robber cannot escape. □

Proposition 3.24

$$\mathrm{bn}(G) \leq \mathrm{cn}(G)$$

Proof. Let $\mathcal{B}$ be a bramble of order $\mathrm{bn}(G)$ and assume there are only $\mathrm{bn}(G) - 1$ cops. Let C be the set of initial positions of the cops. By definition there exists a set $X \in \mathcal{B}$ such that

$X \cap C = \emptyset$. The robber moves to some vertex $x \in X$. After that, the game really begins, a fraction of the cops moves and lands, so that the new set occupied by the cops is C'. Again there exists $X' \in \mathcal{B}$ such that $X' \cap C' = \emptyset$. During their flight the only occupied vertices are $C \cap C'$ so $X \cup X'$ is entirely free of cops, and the robber can freely move from X to Yn and this strategy can be applied for ever. □

3.6 Treewidth and Planar Graphs

To begin this section, let us try to begin a proof of Wagner's Conjecture. Suppose $(G_n)_{n\in\mathbb{N}}$ is a sequence of graphs. We want to prove that there exists G_i and G_j with $i < j$ and G_i is a minor of G_j. If there exists G_i such that G_1 is a minor of G_i, then we have won. If not, we know that $G_2, G_3, \ldots$ all belong to the class of G_1-minor free graphs. If we can prove that this class has bounded treewidth, then we have won by Theorem 3.9. We are going to see that this is true if G_1 is planar. Theorem 3.27 below, states that the class of H-minor free graphs has bounded treewidth if H is planar (and in fact if and only if).

First, note that every planar graph is the minor of some grid

Proposition 3.25

If G is a planar graph, then there exists k such that G is a minor of $G_{k,k}$

Proof. Draw G in the plane and superpose over it a very fine grid. Done □

Therefore, for a given H there exists a $k(H) \in \mathbb{N}$ such that the class of H minor free graphs is included into the class of $G_{k,k}$-free graphs. It remains to prove that these have bounded treewidth. The following theorem, often called the Grid Minor Theorem, achieves this.

Theorem 3.26

For any k, the class of $G_{k,k}$-minor free graphs has bounded treewidth.

Equivalently if a graph has large enough treewidth, it will contain any grid, as a minor. The proof of this result will not be given here, because it is long and difficult, we refer to the reader to [7]. In 2013, Chekuri and Chuzhoy, [5], proved the first polynomial bound, that is there exists a $C > 0$ such that the class of $G_{k,k}$-minor free graphs has treewidth at most k^C.

We are now able to prove the aforementioned result.

Theorem 3.27

The class of H-minor free graphs has bounded treewidth if and only if H is planar

Proof. Suppose first that H is not planar. Since the class of planar graph is minor-closed, this implies that the class of H-minor free graphs contains all planar graphs, and has therefore unbounded treewidth, thanks to Corollary 3.20.

Conversely, if H is planar, then by Proposition 3.25 the class of H minor free graphs is contained in the class of $G_{k,k}$-minor free graphs, for k sufficiently large. But now Theorem 3.26 tells us that this class has bounded treewidth. □

With this result and the discussion at the beginning of this section, we now have.

Corollary 3.28

The class of planar graphs is wqo for the minor relation.

3.7 Wagner's Conjecture

To get the proof of Wagner's conjecture, we would like to imitate the proof of Theorem 3.28. So we start by considering a sequence $(G_n)_{n\in\mathbb{N}}$ that is supposedly a counterexample. As before we can assume that no graph G_i with $i \geq 2$ has G_1 as a minor. And as before we would like to use this fact to get some structure for these graphs. In fact it is sufficient to get a structure theorem for all graphs not containing K_l, for l fixed as a minor. We already saw two theorems characterising this for $l \leq 5$.

It would be nice to have this for every l, unfortunately, we can guess that it would be very complicated; So the idea is to prove approximate characterisations.

Let us start with a technical definition. If C is a cycle, a **vortex** on C is defined the following way :

- select a collection of arcs $A_1, A_2, \ldots, A_l$ on C so that each vertex is in at most k arcs.
- For each arc we add a vertex v_j that is linked to some vertices of A_i.
- we can also add edges $v_i v_j$ if $A_i \cap A_j \neq \emptyset$.

Now let us define a class $\mathcal{G}_k$

i Start with a surface of genus at most k with a graph G embedded in it so that each face is homeomorphic to a disc.

ii Add at most k **vortices** on faces of G

iii Add at most k vertices (apexes) linked arbitrarily to the rest of the graph.

Such are graphs are called "nearly embeddable" in a surface of genus k. Eventually define $\mathcal{L}_k$ as the closure (for clique sums) of the class $\mathcal{G}_k$. Now the fundamental structure theorem can be given.

Theorem 3.29

For every graph H, there exists an integer k such that the class of H-minor free graphs is included in $\mathcal{L}_k$.

Eventually this says that H-minor free graphs have on the outside a tree like structure (due to the sequence of clique sums operations), and inside these are graphs of bounded genus plus a bounded number of perturbations (the vortices and apex vertices). Very (very) roughly, graphs of bounded genus are taken care of by induction on the genus, and then Kruskal's Theorem's proof is adapted to deal with the tree structure of clique sums operations.

3.8 Rooted disjoint path problem and minor detection

Several times in this course we have seen the following connectivity problem :
Input : A graph G, an integer k and two subsets of vertices X and Y be two subsets of vertices of size k
Output : TRUE if there exists k vertex disjoint paths from A to B?
As we have seen, this problem of connectivity has got a dual by Menger's theorem in terms of separators (and hence is in co-NP) but is in fact polynomially solvable by max-flow techniques.

Nevertheless, the smilingly similar problem is NP-complete for $k \geq 2$:
Input : A graph G, an integer k, and two subsets of vertices $X = \{x_1, x_2, \ldots, x_k\}$ and $Y = \{y_1, y_2, \ldots, y_k\}$
Output : TRUE if there exists disjoint paths $P_1, P_2, \ldots, P_k$, such that P_i is a path from x_i to y_i.

In the Graph Minor series of papers, Robertson and Seymour gave a polynomial algorithm for fixed k, and as was discussed in Chapter 1,this gives a polynomial algorithm to decide if a fixed H is a minor of some input graph G.

Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem mentioned above. One is the following : in some situations it is possible to prove that there exists vertex v which is **irrelevant** to the existence of these paths, and if we can find it, we delete it from the graph, and apply the algorithm inductively.

A simple case is the case where G contains a big clique. Suppose it contains a clique of size $2k$. Then apply Menger to $X \cup Y$ and C, if we can find the $2k$ disjoint paths we can answer YES by rerouting inside the clique. If not, there is a separator S of size $s < 2k$ and in fact there exist a set of paths $\mathcal{P}$ from $X \cup Y$ to C of size s. By considering minimal such paths we know that there exists a subset C' of C which don't appear on these paths. Now if there exists k-disjoint routed paths from X to Y, then by rerouting them using the paths in $\mathcal{P}$ we know that there also exists a solution which avoids C'. These vertices are *irrelevant* to our problem. So we can delete those vertices and try again, and thus we can assume that the graph does not contains a big clique. What Robertson and Seymour prove is that it works similarly if G contains a very large grid minor. If the desired paths exists, there is surely a way to reroute the parts of the paths that go through the grid so that some vertex in the 'middle' of the grid is not used, and this will be our irrelevant vertex.

Thanks to Theorem 3.26 we know that if the treewidth is large enough, such a big grid minor exists. If not, there we are in the case of bounded treewidth, and we will see in the next chapter that dynamic programming approaches permits us tho solve the problem. So the difficult part is to effectively and efficiently find the irrelevant vertex in the case of big treewidth. This splits into two parts : first if the graph contains a large clique minor, this is not too difficult, then if no large clique minor exists, Robertson and Seymour use their structure Theorem 3.29 and the existence of a large grid minor to find the irrelevant vertex.

As mentioned in the first chapter, this theorem implies Theorem 1.11 saying that minor detection is polynomial, and hence it implies that any closed under minors property can be decided in polynomial time, since such a property is defined by a finite list of excluded minors. For example this theorem solves the question for linklessly embeddable graphs (graphs that can be embedded in three-dimensional spaces, so that two cycles can always be pulled apart, that is they don't entwine like links in a chain), a problem that was open before the Graph Minors papers.

Chapter 4

Tree Width and Algorithms

4.1 FPT Algorithms

A parametrized algorithmic problem is a problem where a certain parameter is given in addition to the input. We usually denote by k the parameter and n the size of the input. There are roughly three possibilities for a parametrized algorithmic problem.

- Either the problem is already hard for fixed k.
  Example = compute $\chi(G)$ with parameter the solution. NP hard for $k = 3$.
- Or the problem is polynomial of k fixed.
  Example = Decide if $\alpha(G) \leq k$ with parameter k needs exhaustive search : $O\left(n^k\right)$.
- FPT : Algorithm in time $O\left(f(k)n^{O(1)}\right)$

The parameter k can be directly the size of the solution, or an implicit parameter of the input graph (like the diameter, maximum degree, the treewidth).

One easy and classic example that we will use a lot as a common thread throughout this section is the **k-Vertex Cover Problem (k-VC)** : decide whether there is a set of at most k vertices that is incident to every edge of the graph. Even though a vertex cover is the complement of an independent set, note that a FPT algorithm parametrized by the size of the solution for the first problem does not give one for the second (in fact no $n^{o(k)}$ is known for max independent set).

In this chapter, we will prove two kinds of FPT algorithms :

- When the parameter is the treewidth : in that case we will see that many classical NP-hard problems admit FPT algorithms. For that we will need to explain how to compute (in FPT time) an optimal or near optimal tree decomposition. We will also discuss rapidly the connection with logic via the famous theorem of Courcelle.
- Second is when the parameter is classically the size of the solution. Here several aspects will be explored : first, as mentioned in the introduction we will explain why the general theory of Graph Minors of Robertson and Seymour allow us to guarantee that some problems admits FPT algorithms. And second we will see what are called WIN-WIN approaches, that is roughly when we can obtain FPT algorithms parametrized by size of the solution if we

know that one is known when the parameter is the treewidth. Finally we will show some particular methods to improve the complexity when the input graphs are planar (thanks to the Grid Minor Theorem).

The chapter is not exactly divided as suggested by the two items above : we will start with the consequence of Robertson and Seymour Theorems (that is with parameter size of the solution), then discuss all the results about bounded treewidth, and then come back to Win-Win approaches at the end.

4.2 Existence of FPT via Robertson and Seymour Theorems

Assume C is a class of graphs, and let us look at the following algorithmic question : Can we find a $f(k)Poly(n)$-time algorithm to decide if in a graph G on n vertices, there exists $X \subset V(G)$ with $|X| \leq k$ and $G \setminus X \in C$? Note that if the class C is minor closed, then the answer is YES, there exists such an algorithm, since the set of YES instances for this problem is still minor closed (why?). Hence by Robertson and Seymour theorems, there exists a finite list of obstructions, and thus a $O(n^3)$ algorithm (where the constants depends on k)

Note that k-Vertex Cover corresponds exactly to this problem for being the class of stable sets. Similarly, if we take C to be the set of forests, we get the classical problem of deciding whether it is possible to intersect all cycles of a graph with less than k-vertices (k-Feedback Vertex Set Problem).

The problem with this answer is that it is just existential since we do not know how to get the bounds (Robertson and Seymour theorems are not constructive).

.

4.3 Algorithms on Trees

We are going to start with a short section about dynamic programming on trees. Let us consider the problem of maximum weight independent set. The input is a graph $G = (V, E)$ whose vertices are given weights through a function $\omega : V \to \mathbb{R}$. We define the weight of a subset of vertices to be the sum of the weights of its elements. The problem is now to find an independent set of maximum weight. Of course the problem for general graphs is NP-complete (it contains the maximum independent set problem). Nevertheless for trees, there exists an easy polynomial algorithm for this problem.

Fix a root r arbitrarily and orient all edges away from this root. Denote by $C(v)$ the set of children of v, by $T(v)$ the subtree rooted at a vertex v (hence $T(r) = T$) and by

- $W(v)$ denote the maximum weight of an independent set of $T(v)$,
- $W^+(v)$ denote the maximum weight of an independent set of $T(v)$ containing v
- $W^-(v)$ denote the maximum weight of an independent set of $T(v)$ not containing v

Then we can compute inductively W, W^+, and W^-, since

$$\begin{aligned} W(v) &= \max\left(W^+(v), W^-(v)\right) \\ W^+(v) &= w(v) + \sum_{u \in C(v)} W^-(u) \\ W^-(v) &= \sum_{u \in C(v)} W(u) \end{aligned}$$

Eventually, we can compute $W(r)$ which is exactly the value of the maximum weight independent set of T.

Like this problem, many NP-complete problems become polynomial on trees. On the next section we are going to see that this stands also for bounded treewidth graphs. That is we are going to find algorithms whose complexity is a function $f(t, n)$ of the treewidth k and the size of graph n, and this function will be a polynomial in n.

4.4 Algorithms for bounded-treewidth graphs

In this section we will assume that the graph G has treewidth at most t, AND we will assume that we are given a nice tree decomposition T of width at most t (we will explain later how to get rid of this second assumption later, but is important that we do not take care of the time complexity needed to compute this decomposition). As we will see through examples, many NP-hard problems become polynomial when restricted to bounded treewidth graphs.

Theorem 4.1

Computing maximum weight independent set can be solved in time $O(f(t).n)$

Proof. Let (T, W) be an tree decomposition of width $k = tw(G)$, and root T at some arbitrary vertex. Now for any vertex t of T, define G_t the subgraph of G induced by vertices that belong to bags $W_{t'}$ where t' is a descendent of t. Now for any stable set S which is a subset of W_t, define

$$\phi(S, t) = max\{|S'|, S' \text{ stable set of } G_t \text{ such that } S' \cap W_t = S\}.$$

If t is a leaf of T then $\phi(S, t) = |S|$. Assume now that t has children $t_1, \ldots, t_p$, and assume the value $\phi(S, t_i)$ is known for any i and stable set inside W_{t_i}. We prove that we can compute $\phi(S, t)$ for any stable set $S \subset W_t$. The important and easy fact is that a set S' is a stable set of G_t if and only if its intersection with W_t is a stable set, as well as all its intersections with the graphs G_{t_i}. Therefore for any i let S_i be a stable set of W_{t_i} such that $S \cap W_{t_i} = S_i \cap W_t$ and $\phi(S_i, t_i)$ maximal. To find such a set we need to explore all stable sets of W_{t_i}, which takes time $O\left(2^k\right)$ for every i. Moreover we have that

$$\phi(S, t) = |S| + \sum_i (\phi(S_i, t_i) - |S_i \cap W_{t_i}|).$$

The total computing time is therefore $\sum_{t \in V(T)} 2^k 2^k k^2 p_t$ where p_t is the number of children of node t. By using a tree decomposition with a linear number of nodes we get a complexity of $O\left(4^k k^2.n\right)$

□

Theorem 4.2

Decide if G is 3 colourable can be solved in linear time.

Proof. Let X_1 be any bag of T, and fix it as a root of T. Denote by T_i the trees rooted at vertex X_i. For each i denote by C_i the set of all 3 colourations of $G_{|X_i}$ that can be extended to 3 colourations of T_i. If we manage to compute all C_i, then we conclude by checking if C_1 is non empty.

For every leaf X_i computing C_i can be done exhaustively in time $O(3^t)$. for every internal node X_i whose children have been dealt with, we compute all 3 colourations of $G_{|X_i}$ to get a set C_i'. Then for each colouration c in this set, we keep it in C_i if for every children X_j, there exists a coloration c_j such that it coincides with c when restricted to $X_i \cap X_j$. It takes time $3^t.3^t.p_i.t^2$ if p_i is the number of children of X_i.

In total this gives $\sum_{X_i} 9^t.t^2(p_i + 1) = O\left(9^t t^2.n\right)$

□

Often it is useful to design those algorithms to work with a tree decomposition that has nice structural properties. We define those below.

Definition 4.3

*A **nice tree decomposition** of G is a tree decomposition where T is a rooted binary tree with bags $(W_t)_{t\in V(T)}$ and each inner node t is of three possible kind :*

- *Join : t has two children t_1 and t_2 and $W_t = W_{t_1} = W_{t_2}$.*
- *Introduce : t has one children t' and $W_t = W_{t'} \cup \{x\}$ where $x \notin W_{t'}$*
- *Forget : t has one children t' and $W_{t'} = W_t \cup \{x\}$ where $x \notin W_t$*

The important fact is that for every graph one can find an optimal tree decomposition that is simple.

Proposition 4.4

Given a tree decomposition of width w and with n nodes, it is possible in time $O\left(w^{O(1)}n\right)$ to transform it into a simple one with width w and with $O(nw)$ nodes.

Proof. Root the decomposition arbitrarily. For each internal node with p children, it is possible to add $2p$ new join nodes to make it binary. Then for each node t with one children t'. It possible now to replace the edge tt' by a path of at most $2w$ forget or introduce nodes. □

Using this kind of special decomposition often makes it easier to design dynamical programming algorithms using tree decompositions. See Exercice **??**

Let us illustrate how to use these decomposition to solve the Vertex Cover problem. A set of vertices S is a **vertex cover** if every edge is incident to a vertex in S (or equivalently if $V \setminus S$ is an independent set). We have seen an algorithm for this problem (since we have seen one for max independent set), here we are giving another one using these simple decompositions.

Theorem 4.5

If G is given with a tree decomposition of width t, one can find the minimum size of a vertex cover in time $2^t.O(n)$.

Proof. Let (T, W) be a nice tree decomposition of G. For every node t of T, denote by G_t the subgraph of G induced by the vertices that belong to a bag that is a descendent of t. Now for every C that is a subset of W_t, let $\phi(t, X)$ be equal to the minimum size of a vertex cover X' of G_t such that $C' \cap W_t = X$ if such a X' exists, and ∞ otherwise.

Now we will show how to compute $\phi(t, X)$ for all t and X using a bottom up approach. We have to deal with the leaves of the tree and then with the three possible cases for internal nodes (Forget, Introduce, Join).

- if t is a Leaf node of T, with $W_t = \{x\}$. Then $\phi(t, \{x\}) = 1$ and $\phi(t, \emptyset) = 0$
- if t is a Forget node with child t' and $W_t = W_{t'} \setminus \{x\}$, then for all $X \subset W_t$,

$$\phi(t, x) = min\{\phi(t', X), \phi(t', X \cup \{x\})\}$$

- if t is an Introduce Node with child t' and $W_{t'} = W_t \setminus \{x\}$, then for all $X \subset W_t$:
  1. If X is not a vertex cover of the graph induced by $W_{t'}$, then $\phi(t, X) = \infty$.
  2. If X is a vertex cover of the graph induced by $W_{t'}$, et $x \in X$, then $\phi(t, X) = \phi(t', X \setminus \{x\}) + 1$.
  3. If X is a vertex cover of the graph induced by $W_{t'}$, et $x \notin X$, then $\phi(t, X) = \phi(t', X)$.
- if t is an Join Node with child t' and t'', then for all $X \subset W_t$, $\phi(t, X) = \phi(r, X) + \phi_{t''}(X) - |X|$.

These results are not difficult to prove, and therefore all values of $\phi(t, C)$ can be computed in time proportional to $|V(T)|2^{1+\mathrm{tw}(G)}$. It is clear that the min Vertex cover is given by the minimum ever C of all $\phi(r, C)$, where r is the root of T, which gives the algorithm.

□

Here is a list of results one can prove similarly using a tree decomposition of treewidth k.

Theorem 4.6

Let G be given with a tree decomposition of width at most k.

1. *Computing $\chi(G)$ can be done in time $O(f(k).n)$*
2. *Computing $\omega(G)$ can be done in time $O\left(2^k.k^k.n\right)$*
3. *Computing $\gamma(G) := \min\{|X|, X \text{ dominating set }\}$, can be done in time $O(f(k).n)$.*
4. *Deciding if G has a hamilton cycle can be done in time $O(f(k).n)$*

4.5 Monadic Second Order Logic

A celebrated algorithmic meta-theorem of Courcelle ([6]) generalizes all the previous results to monadic second order formulas. Let us explain briefly what these are. We are interested into logical formulas on the graph seen as a relational structure. This means variables are vertices or edges, and one can build formulas inductively using :

- atomic formulas : $x = y$, $v \in X$, $e \in F$ for subsets of vertices or edges.
- the binary relation xIe which is satisfied if $x \in V$ and x is incident with $e \in E$.
- logical operators $\vee$ and $\wedge$ and $\neg$
- quantifiers $\forall$ and $\exists$

To be precise about quantifiers, first order formulas is the fragment of logic where one is allowed to use quantifiers over vertices and edges ($\forall v \phi(v)$), MSO_1 is the fragment where one is allowed in addition to quantify over sets of vertices, and eventually MSO_2 is a larger fragment where in addition one can quantify over sets of edges.

Every time one property is written as such a formula, one can give it a name and use it as a macro to produce other formulas (the length of the formula of course should take this into account). For example $x \neq y$ instead of $\neg(x = y)$ or $X \subset Y$ instead of $\forall x \in V(x \in X \Rightarrow x \in Y)$.

In a formula, a variable that is not in the scope of a quantifier is called a free variable. For example the property $adj(x, y)$ that vertices x and y are adjacent has two free variables and can be written

$$(x \in V) \wedge (y \in V) \wedge (x \neq y) \wedge (\exists e \in E xIe \wedge yIe)$$

and the fact that the graph contains a vertex adjacent to all others is formulated by

$$\exists x \in V(\forall y \in V\ (y \neq x) \Rightarrow adj(x, y))$$

These were first order formulas, with second order one can express for example 3 colourability :

$$\begin{aligned} Col3 := \quad & \exists X_1 \subset V\ \exists X_2 \subset V\ \exists X_3 \subset V \\ (\forall x \in V \quad & (x \in X_1 \vee x \in X_2 \vee x \in X_3) \\ & \wedge \neg(x \in X_1 \wedge x \in X_2) \wedge \neg(x \in X_1 \wedge x \in X_3) \wedge \neg(x \in X_2 \wedge x \in X_3)) \\ \wedge (\forall xy \in E \quad & \neg(x \in X_1 \wedge y \in X_1) \wedge \neg(x \in X_2 \wedge y \in X_2) \wedge \neg(x \in X_3 \wedge y \in X_3)) \end{aligned}$$

The theorem of Courcelle asserts that every such property is easy to decide for bounded treewidth graphs.

Theorem 4.7

Let ϕ be a MSO_2 formula of length k, and let G be a graph on n vertices with $\mathrm{tw}(G) \leq t$. There exists an algorithm that performs in time $f(k, t).O(n)$ to decide whether G satisfies ϕ.

Note that the formula that says that G has a vertex cover of size at most k has length $O(k)$.

$$\exists x_1 \exists x_2 \ldots \exists x_k \; \forall e \in E \; (x_1 Ie \vee x_2 Ie \vee \ldots \vee x_k Ie)$$

So applying the previous theorem would only give a $f(k,w).O(n)$ algorithm to decide if a graph of treewidth at most t has a vertex cover of size at most k.

There is a version of Courcelle Theorem for optimisation problems instead of decision ones.

Theorem 4.8

Let $\phi(S)$ be a MSO_2 formula of length k with free variable S, and let G be a graph on n vertices with $\mathrm{tw}(G) \leq t$. *There exists an algorithm that performs in time $f(k,t).O(n)$ that can find the minimum size of an S that satisfies $\phi(S)$.*

(For Vertex Cover the formula is then just $\forall e \in E \; \exists x \in S \; xIe$.)

This is an example of such a theorem, more complicated versions exist (with several free variables for example).

Courcelle's Theorem is a powerful tool for classification but has limitations because there are no good estimates on the running time. So in order to get practical algorithms, one still needs to design algorithms for each problem.

4.6 Computing tree width

Until now, we have delayed the problem of finding a small width decomposition. The main problem with computing treewidth is that the following problem is NP-complete ([2]):
Input : G, t
Output : TRUE if $\mathrm{tw}(G) \leq t$

(It is interesting to mention here that it is still open whether the same problem is polynomial for planar graphs.)

On the other hand, since we are interested into using tree decompositions to devise FPT algorithms with parameter $\mathrm{tw}(G)$, we would be happy to compute such a tree decomposition in time $O(f(\mathrm{tw}(G)).P(n))$. There exists such algorithms (even linear in n, see [3] for a $O\left(2^{\mathrm{tw}(G)^2} n\right)$ algorithm), but in the sole purpose of having FPT algorithms, we need in fact only something weaker, such as the following.

Theorem 4.9

There exists an algorithm with input a graph G and an integer k and that outputs in time $O\left(f(k).n^2\right)$:

- *either* $\mathrm{tw}(G) \geq k$
- *or a tree decomposition of width at most* $4k-1$.

Note that this is enough to apply the algorithms described in the previous section for bounded treewidth graphs. Indeed by applying the previous theorem for $k = 1$, $k = 2$, $k = 3$,... one is guaranteed to find a tree decomposition of G of width at most $4\text{tw}(G)$ in time $O\Big(f(\text{tw}(G)).n^2\Big)$.

Before proving the theorem let us define the notion of **good separator** for a set W of vertices. S is a good separator for W if S disconnects G into non-trivial subsets V_1 and V_2 such that for $i = 1, 2$, V_i contains at most $2|W|/3$ vertices of W (and therefore at least $|W|/3$ - these are often called 1/3-2/3 separators). One can prove that if $\text{tw}(G) < k$, every X of size at least $2k + 1$ admits a good separator of size at most k (Exercise **??**). The main ingredient for the proof is the fact that the converse is almost true.

Proof.

We prove inductively that there exists an algorithm for the following (it suffices to apply it with $W = \emptyset$ to get the theorem).

Input : G, $W \subset V(G)$ such that $|W| \leq 3k$

Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k - 1$ where $W \subset root(G)$

If G has less than $4k$ vertices then put all vertices in a single bag.

If not, but W has less than $2k + 1$ vertices, then augment W arbitrarily by adding vertices until its size is at least $2k + 1$. If now W admits no good separator of size at most k, than by what precedes it is a certificate than $tw(G) > k$. Assume for the moment that it exists and we are able to compute it. We are given S such that $G \setminus S$ is the disjoint union of G_1 and G_2 with $W \cap V(G_i) \leq 2|W|/3$. Define $W_i = S \cup (W \cap V(G_i))$. Then $|W_i| \leq k + \frac{2}{3}3k = 3k$ and we can inductively apply the algorithm on (G_i, W_i) for $i = 1, 2$ to get either a certificate that $tw(G) \geq k$ or two decompositions T_1, T_2 of G_1 and G_2. It suffices to add a root bag containing all vertices in $W \cup S$ attached to the roots of T_1 and T_2 to get the desired tree decomposition (note that $W \cup S - 1 \leq 4k - 1$)

The only thing remaining is to prove that we have an algorithm to find a good separator for W one exists. Such a set exists if and only if one can partition W into three subsets W_1, W_2, W_0 such that W_1 and W_2 have size at most $2|W|/3$, and W_0 is a subset of a separator of size at most k separating V_1 and V_2 where $W_i \subset V_i$. Observe that W_0 can be extended into such a separator if and only if in $G \setminus W_0$, there are at most $k - |W_0|$ disjoint paths from W_1 to W_2. This can be tested by a max flow technique in time $O\Big(k^2n\Big)$ (because the graph has at most kn edges, otherwise it cannot have treewidth at most k). If the answer is no for every partition there the set W does not have a good separator. Otherwise we find the separator. There are less than 3^{3k}ways of defining the partition W_0, W_1, W_2 so this gives complexity $O\Big(27^k.k^2n\Big)$ for this step and therefore $O\Big(27^k.k^2n^2\Big)$ in total since the tree decomposition has at most n nodes.

□

As mentioned before, the proof above contains the following result

Theorem 4.10

Let G be a graph such that every $X \subset V(G)$ of size at least $2k + 1$ admits a good separator of size at most k, then $tw(G) \leq 4k - 1$.

which gives a proof of the approximation of the duality theorem since if X has size $2k + 1$ the

set of connected subgraphs containing at least $k + 1$ vertices of X forms a bramble.

Theorem 4.11

For any graph G

$$tw(G) \leq 4bn(G) - 1$$

4.7 Win-Win Approach an Planar Problems

4.7.1 Win-Win

How can we use the fact that we have good algorithms for bounded treewidth graphs (either by Courcelle's Theorem, or by dynamical programming if we seek practical ones) to design algorithms for all graphs?

Let us take the example again of k-Vertex Cover. Remember that for a graph G we are able in time :

- $f_1(k)O(n)$ to decide whether $\text{tw}(G) \leq k$ and find a tree decomposition of width k in that case.
- $f_2(k)O(n)$ to decide G has a k-VC if we are given a tree decomposition of width k of G.

The missing part is in fact trivial : for any graph G, if G has treewidth larger than k, then it has no k-VC. (why?)

This gives a scheme for designing FPT algorithms as long as the problem we are solving has some property that implies that there is a bound t such that if $\text{tw}(G) > t$ then either the answer is always YES or always NO.

4.7.2 Bidemnsionality - the case of planar graphs

By dynamic programming, or Courcelle's Theorem, we can get an $2^{O(tw(G))}O(n)$ algorithm. This gives an FPT parametrized by treewidth but also by the size of the solution, since $minVC \geq tw$ (why?).

Let us look at planar graph to illustrate a concept called *bidimensionality*. Here are a few observations :

- If H is a minor of G, then $\min VC(H) \leq \min VC(G)$.
- If the treewidth of a **planar** G is larger than $4k$, it contains a grid $G_{k,k}$ as a minor.
- The grid $G_{k,k}$ has no vertex cover of size at most $k(k-1)/2$

This implies an FPT algorithm for k-VC in planar graph in time $2^{O(\sqrt{k})}.n^c$: If $tw > 4\sqrt{2k+1}$ one can answer NO, otherwise we do dynamic programming. This type of approach works as long as the problem satisfies the following : the answer is trivially YES or NO if the graph contains a large enough grid.

Bibliography

[1] D. Archdeacon and P. Huneke. A kuratowski theorem for nonorientable surfaces. *Journal of Combinatorial Theory, Series B*, 46(2):173–231, 1989.

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

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

[4] J.A. Bondy and U.S.R Murty. Graph theory, ser. *Graduate Texts in Mathematics. New York: Springer*, 244, 2008.

[5] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 60–69, New York, NY, USA, 2014. ACM.

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

[7] R. Diestel. Graph theory,(graduate texts in mathematics, vol. 173). *Recherche*, 67:02, 2005.

[8] Rod G Downey and Michael Ralph Fellows. *Parameterized complexity*, volume 3. springer Heidelberg, 1999.

[9] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 2(3):326–336, 1952.

[10] R.M. Karp. Reducibility among combinatorial problems. *50 Years of Integer Programming 1958-2008*, pages 219–241, 2010.

[11] László Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86, 2006.

[12] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math*, 10(95-115), 1927.

[13] Neil Robertson and P.D. Seymour. Graph minors. VII. disjoint paths on a surface. *Journal of Combinatorial Theory, Series B*, 45(2):212–254, 1988.

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

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

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

[17] P.D. Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.