Minors - Treewidth - Algorithms
MPRI Graph Algorithms

Pierre Charbit - charbit@irif.fr
Graph theory

Graphs: a mathematical object and an efficient modeling tool.

Important questions:

- What classes of graphs have good algorithmic properties? (colouring, clique max...)
- What classes of graphs have good structural properties? (decomposition theorem, elimination ordering...)

Forbidding a substructure:

- Minors: Robertson and Seymour, 1983-2012
- Topological minors
- Induced subgraphs
Basic Definitions and Terminology

In this course, all graphs are simple (no parallel edges) and without loop.

If \( G \) is a graph, we denote \( V(G) \) its set of vertices and \( E(G) \) its set of edges.

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 denoted \( \Delta(G) \).

A graph with no edge is **stable set**, or **independent set**, and a graph with all possible edges (\( \binom{n}{2} \)) is a **clique**, or **complete graph**. The complete graph on \( n \) vertices is denoted \( K_n \). The complete bipartite graph with parts of size \( a \) and \( b \) is denoted \( K_{a,b} \).

The **path** \( P_k \) is a graph with \( V(P_k) = \{x_1, x_2, \ldots, x_k\} \), with edges \( E(P_k) = \{x_ix_{i+1}, \ 1 \leq i \leq k - 1\} \). The vertices \( x_1 \) and \( x_k \) are called the **endpoints** of the path. If we add the edge \( x_kx_1 \) to \( P_k \) then the resulting graph is the **cycle** on \( k \) vertices, denoted \( C_k \).
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?

**CLASSIC**: Can be solved in time $O(k|E(G)|)$ (variant of Ford-Fulkerson Algorithm)

The maximum value $k$ corresponds to a minimum vertex cut separating A and B and is a classical result of Menger.

**Theorem 1** (Menger, 1927, [6])

*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.*
A seemingly 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 $y_i$.

- Related to commodity flow problem, has many applications (VLSI design)
- With $k \geq 2$ 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 $k$.

**Theorem 2 (Robertson-Seymour, [7])**

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

(improved to quadratic time by Kawabayarashi, Kobashi and Reed)
**Definition**

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.

**Exercise 1**

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

- With $H$ part of the input: NP-complete
- With $H$ fixed, polynomial thanks to the $k$-disjoint path problem algorithm (see next slide).
  - Easy algorithm: $O(f(k)n^k)$, where $k = |V(H)|$, and $n = |V(G)|$. Therefore polynomial for every fixed $k$.
  - In 2010, Grohe, Kawabaryashi, Marx, and Wollan proved $O(f(k)n^3)$. Such an algorithm is called Fixed Parameter Tractable (FPT) algorithm (see later in the course)
Topological Minor Detection: Polynomiality when $H$ is fixed

**Theorem 3**

Let $H$ be a fixed graph with $k$ edges. One can decide whether $H$ is a topological minor of a given graph $G$ in time $O(f(k)n^k)$.

**Sketch proof:**
Let $f : V(H) \to V(G)$ be an injection.
Observe that there is $\binom{n}{|V(H)|}$ 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.
Consequences

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.

Example of such class?

**Theorem 4 (Kuratowski, 1930, [5])**

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

This is just an example: of course one does not need Robertson and Seymour Theorem to get polytime algorithms for recognising planar graphs (there exist even linear algorithms to do that)
3 - Minors
We define three operations on a graph $G$

1. **Remove a vertex** $v$ (and all its incident edges), denoted $G \setminus v$.

2. **Remove an edge** $e$ (but not its end vertices), denoted $G \setminus e$.

3. **Contract an edge** $e = xy$, denoted $G/e$:
   
   (i.e. remove $x$ and $y$, add a new vertex $z$ with neighbourhood $N(z) = (N(x) \cup N(u)) \setminus \{z\}$ (no loops))

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**

Let $G$ and $H$ be two graphs.

- $H$ **induced subgraph** of $G$ if $H$ obtained from $G$ by the repeated use of 1.
- $H$ **subgraph** of $G$ if $H$ obtained from $G$ by the repeated use of 1 and 2.
- $H$ **spanning subgraph** of $G$ if $H$ obtained from $G$ by the repeated use of 2.
- $H$ **minor** of $G$ if $H$ obtained from $G$ by the repeated use of rule 1,2 and 3.
- $H$ **topological minor** of $G$ if $H$ is a minor of $G$ and every contraction used was topological.
Here is an equivalent definition for minors that is often useful.

**Lemma**

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 $v_i v_j$ of $H$, there exists an edge between $G_i$ and $G_j$. 
Exercises

**Exercice 2**
Prove that a graph $G$ is a forest if and only if it does not contain $C_3$ as a minor.

**Exercice 3**
Show that the $(3 \times 3)$-grid has a $K_4$-minor

![Grid graph 3x3](image)

**Exercice 4**
1. Prove that every graph with average degree at least $2^{r-2}$ contains $K_r$ as a minor.
2. For $r$ fixed, does there exist $K_r$ minor free graphs with arbitrarily large chromatic number?
Minors Vs Topological Minors

- By definition $H$ topological minor of $G \Rightarrow H$ minor of $G$
- converse not true: **EXERCISE**

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

**Theorem 5**

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:

- Assume $H$ minor of $G$
- Take $G'$ topological minor of $G$ minimal (number of edges) such that $H$ minor of $G$.
- Use previous lemma.
- Each $G_i$ is a tree with at most 3 leaves and no vertex of degree 2
- Each such tree must be a star, so we get the topological minor
Similar argument proves this more general result.

**Theorem 6**

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.

This result combined with the theorem on topological minor detection now clearly implies the following theorems due to Robertson and Seymour. (Graph Minors XIII [8])

**Theorem 7** *(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 $G$.

**Corollary**

If $C$ is a class of graphs defined by forbidding finitely many minors, then there exists a polynomial algorithm to decide whether an input graph belongs to $C$. 
**Wagner Conjecture**

**QUESTION**

What are the families defined by finitely many forbidden minors?

Examples:

<table>
<thead>
<tr>
<th>Graph Class</th>
<th>Obstructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forests</td>
<td>triangle $K_3$</td>
</tr>
<tr>
<td>Union of Paths</td>
<td>triangle, claw $K_{1,3}$</td>
</tr>
<tr>
<td>Planar</td>
<td>$K_5$, $K_{3,3}$</td>
</tr>
<tr>
<td>Toric</td>
<td>$\geq 16629$ (but finite)</td>
</tr>
</tbody>
</table>
**Question**

What are the families defined by finitely many forbidden minors?

- 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 (>700 pages of proof (Graph Minors I-II-III .... XXV)), solving a conjecture of Wagner from 1937 says that this is sufficient.

**Theorem 8** *(Graph Minor Theorem, [9])*

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

With an important consequence

**Corollary**

*If $C$ is a minor closed class, then there exists a polynomial to decide if an input graph belongs to $C$.*
Exercises

**Exercice 5**
For each of the following classes, decide if it is minor closed or not. If not, try to describe the smallest minor closed class containing it: cliques, paths, cycles, graphs of max degree $k$?

**Exercice 6**
Prove that the following problems are solvable in time $O(f(k)) \cdot n^3$.

- **$k$-Feedback vertex set**
  **Input**: A graph $G$.
  **Output**: TRUE if there exists $k$ vertices in $G$ that intersect every cycle of $G$.

- **$k$-Vertex Cover**
  **Input**: A graph $G$.
  **Output**: TRUE if there exists a set $S$ of at most $k$ vertices in $G$ such that every edge of $G$ is incident to at least one vertex of $S$.

- **$k$-leaf Spanning Tree**
  **Input**: A graph $G$.
  **Output**: TRUE if there exists in $G$ a spanning tree $T$ with at least $k$ leaves.
4 - Well Quasi Orders
Our goal in this last chapter is a single theorem, one which dwarfs any other result in graph theory and may doubtless be counted among the deepest theorems that mathematics has to offer: in every infinite set of graphs there are two such that one is a minor of the other. This graph minor theorem (or minor theorem for short), inconspicuous though it may look at first glance, has made a fundamental impact both outside graph theory and within. Its proof, due to Neil Robertson and Paul Seymour, takes well over 500 pages.

Reinhart Diestel
**Definition (Bounds)**

For a given minor closed class $C$, a graph $H$ is said to be a **bound** if $H$ is not in $C$ but every strict minor of $H$ is.

**Proposition**

*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 $\mathcal{B}$ as a minor.*

- So Wagner’s conjecture is to prove that a set of bounds is always finite.
- No that by definition the set of bounds forms **antichain** of the minor partial order: no bound is a minor of another bound.
- Is is true that **every antichain is finite**?
**Proposition**

The following are equivalent:

- Every minor closed class has a finite set of bounds
- There is no infinite antichain for the minor relation

**Definition**

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

Wagner’s conjecture is equivalent to say that **the class of all graphs with the minor relation is a wqo**
Exercice 7

For each of these, say if it is a wqo.

- $\mathbb{N} = (\mathbb{N}, \leq)$,
- $\mathbb{R} = (\mathbb{R}, \leq)$,
- $\mathbb{N}^2 = (\mathbb{N}^2, \leq)$ where $(x, y) \leq (x', y')$ iff $x \leq x'$ and $y \leq y'$,
- $\mathcal{G} = (\mathcal{G}, \leq)$ where $\mathcal{G}$ is the class of all graphs, and $G \leq H$ if $G$ (induced) subgraph of $H$
- Finite trees ordered by subgraph relation.
- $\mathcal{G} = (\mathcal{G}, \leq)$ where $G \leq H$ if $G$ topological minor of $H$
The objective of this section is to prove that trees satisfy Wagner’s conjecture.

**Theorem 9 (Kruskal 1960)***

The finite trees are WQO by the topological minor relation, i.e. for every infinite sequence of trees $T_0, T_1, \ldots$, there exists $i < j$ such that $T_i \preceq T_j$.

We will first expose some tools about WQO, then some easier cases before proving the theorem.
Let us start with a Ramsey-type result.

**Proposition**

Let \((X, \leq)\) 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.

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

**Corollary**

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

1. \((X, \leq)\) 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 \(x_i \leq x_j\).

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. If $X$ is a set (the alphabet) $X^*$ denotes the set of finite words (ordered sequences) over $X$.

**Proposition**
Assume $X$ is a finite set. Define a partial order $\preceq$ on $X^*$ by $u \preceq v$ if $u$ is a subword of $v$ (u can be obtained from v by deletion of letters). Then $(X^*, \preceq)$ is a wqo.

\[
abaac \preceq accbaabababbcaaa
\]

**Corollary**
Every Language that is closed for subword is rational.
Proof sketch

Proof by induction on the size of the alphabet $X$

- Assume by contradiction $w_1, w_2, \ldots$ is a bad sequence
- In particular $w_1 \not\leq w_i$ for every $i \geq 2$ !! KEY PART: get some info/structure
- Let $w_1 = x_1 x_2 \ldots x_k$ and define the function $f : X^{(\mathbb{N})} \to \{0, 1, \ldots, k\}$ by

$$f(w) = \max\{i, \ x_1 x_2 \ldots x_i \leq w\} \quad (0 \text{ if no such } i \text{ exists})$$

- There exists $0 \leq j \leq k - 1$ such that $\{i, \ f(w_i) = j\}$ is infinite
- Extract and assume wlog that $f(w_i) = j$ for all $i \in \mathbb{N}$
- Any word $w$ of the sequence can be written as

$$w = u_1 x_1 u_2 x_2 \ldots u_j x_j u_{j+1}$$

where for all $1 \leq i \leq j + 1$, $x_i \not\leq u_i$
- So we have $j$ different sequences, each on a strictly smaller alphabet.
- One can extract an infinite increasing subsequence, contradiction.
Higman’s Theorems

Higman Theorem replaces the finite alphabet by a WQO one

**Theorem 10** (Higman, 1952, [4])

Let \((X, \preceq)\) be a wqo. Define a partial order on \(X^*\) by \(x_1x_2 \ldots x_k \preceq y_1y_2 \ldots y_l\) if there exists an increasing injection \(f : \{1, \ldots, k\} \rightarrow \{1, \ldots, l\}\) such that for every \(i, x_i \preceq y_f(i)\). Then \((X^*, \preceq)\) 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.

For our purposes we will prove a variant of this one (the proofs are almost identical) that concerns the set of all finite subsets of \(X\), denoted by \(X^{(\omega)}\).

**Theorem 11** (Higman, 1952, [4])

Let \((X, \leq)\) be a wqo. Define a partial order on \(X^{(\omega)}\) by \(A \leq B\) if there is an injective mapping \(f : A \rightarrow B\) such that \(a \leq f(a)\) for all \(a \in A\). Then \(X^{(\omega)}\) is also a wqo.
Proof sketch

• Assume for contradiction that $X^{(\omega)}$ has an infinite antichain.

• We start by constructing a bad sequence $(A_n)_{n \in \mathbb{N}}$ as follows:

  • Assume inductively that $A_i$ has been defined for every $i < n$, and that there exists a bad sequence in $[X]^w$ starting with $A_0, \ldots, A_{n-1}$.

  • Choose $A_n$ such that some bad sequence starts with $(A_0, \ldots, A_{n-1}, A_n)$ and $|A_n|$ is minimum with this property.

  • For each $n$, pick an element $a_n \in A_n$, and set $B_n = A_n \setminus \{a_n\}$.

• Since $X$ is wqo, $(a_n)_{n \in \mathbb{N}}$ has an infinite increasing subsequence $(a_n^i)_{i \in \mathbb{N}}$.

• Now look at sequence $(A_0, \ldots, A_{n_0-1}, B_{n_0}, B_{n_1}, \ldots)$.

• By the minimal choice of $A_n$, it is not a bad sequence.

• Conclude.
A famous result from Kruskal is that finite trees are well quasi ordered for the **topological minor** relation. Here we prove a stronger result.

**Definition**

Let \((X, \preceq)\) be a WQO. Let \(T\) and \(T'\) two rooted trees whose vertices are labelled by \(X\). We say that \(T \preceq T'\) if there exists a subdivision \(T''\) of \(T\) with an isomorphism \(\phi\) between \(T''\) to a subgraph of \(T'\) such that:

- **\(\phi\) preserves the tree order**: if \(x\) is an ancestor of \(y\) in \(T\), then \(\phi(x)\) is an ancestor of \(\phi(y)\) in \(T'\).
- **\(\phi\) preserves the labels**: for every vertex \(x\) of \(T\), \(\text{label}(x) \preceq \text{label}(\phi(x))\)

(If \(|X| = 1\) this is the topological minor relation for rooted trees. Restricted to paths, this is exactly the partial order defined on words in Higman’s Theorem.)

**Theorem 12 (Kruskal)**

The order defined above is a well quasi order on labelled rooted trees.
Proof sketch

- Assume by contradiction there is a bad sequence of trees and construct a minimal bad sequence as for Higman’s Lemma
- Denote by $r_i$ the root of $T_i$, and $U_i$ the family of trees obtained by removing $r_i$ from $T_i$. Let $U$ be the union of all $U_i$.
- We claim that $U$ is well quasi ordered
- Indeed if not, we can find a a bad sequence $U_\phi(i)$ such that for all $i$, $U_\phi(i) \in U_\phi(i)$.
- $T_0, T_1, \ldots, T_{\phi(0)-1}, U_{\phi(0)}, U_{\phi(1)}, U_{\phi(2)} \ldots$. It is easy to see that it is a bad sequence, which contradicts the minimality of the $T_i$.
- So $U$ is WQO so by Higmans Lemma the finite subsets of $U$ is also WQO (for the order defined on subsets as in Higmans Lemma)
- Therefore the sequence $(U_i)_{i \in \mathbb{N}}$ admits an infintie increasing subsequence $(U_\psi(i))_{i \in \mathbb{N}}$
- Look now at the sequence of labels of the roots $r_\psi(i)$
- Since the set of labels is wqo, there exists $i < j$ such that $l(a_\psi(i)) \leq l(a_\psi(j))$. Together with $U_\psi(i) \leq U_\psi(j)$, this gives $T_i \leq T_j$, and our contradiction
5 - TreeWidth
**Definition**

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:

- $\forall u \in V(G), \{t \in V(T) : u \in W_t\}$ induces a connected subgraph (a subtree) of $T$.
- $\forall uv \in E(G)$, there exists $t \in V(T)$ such that $u, v \in W_t$

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 this is not an equivalence, it is possible that $T_u \cap T_v \neq \emptyset$ even if $uv \notin E(G)$)

**Definition**

- The *width* of a tree decomposition $(T, W)$ is $\max_{t \in V(T)}(|W_t| - 1)$
- The *tree width* of a graph $G$, denoted $\text{tw}(G)$, is the minimum width of a tree decomposition of $G$
The original graph $G$

A tree-decomposition of width 3

A tree-decomposition of width 2
Example
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**

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

Apply this to any tree decomposition with the set of subtrees associated to the vertices of a clique.

**Corollary**

$tw(G) \geq \omega(G) - 1$
We can ask the optimal tree-decomposition to satisfy some additional properties

**Proposition**

For every graph $G$, there exists a tree decomposition of width $tw(G)$ 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 $T_u = \{f\}$.

As a consequence we get another easy lower bound on the treewidth of a graph.

**Theorem 13**

In every graph $G$, there exists a vertex of degree at most $tw(G)$, i.e. $\delta(G) \leq tw(G)$.

**Corollary**

$\chi(G) \leq tw(G) + 1$
**Proposition**

Let $G$ be a graph, $v$ a vertex of $G$ and $e$ an edge of $G$.

- $\text{tw}(G \setminus e) \leq \text{tw}(G)$
- $\text{tw}(G \setminus v) \leq \text{tw}(G)$
- $\text{tw}(G/e) \leq \text{tw}(G)$

**Proof:**

- for $G \setminus e$, do nothing
- for $G \setminus v$, just remove $v$ from every bag containing it.
- for $G/e$, where $e = uv$: let $w$ be the new vertex. Add $w$ in every bag containing $u$ or $v$, and delete every occurrence of $u$ and $v$. 
**Proposition**

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

**Corollary**

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.

**Theorem 14**

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

**Corollary**

The class of graphs of treewidth at most $k$ has a finite number of bounds.
So, for every fixed $k$, the class $\{G : \text{tw}(G) \leq k\}$ has a finite number of bounds.

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

**Theorem 15**

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

The first is trivial, the proof for $\text{tw}(G) = 2$ shows the role of separators with treewidth (see next slide)
Proof sketch:

- If $G$ contains $K_4$ as a minor, then $\text{tw}(G) \geq \text{tw}(K_4) = 3$.
- If $G$ has no $K_4$ minor, let us prove that $\text{tw}(G) \leq 2$. We proceed by induction on $V(G)$.
- Prove first that every 3-connected graph contains $K_4$ as a minor (Use Menger Theorem).
- So we may assume that $G$ has a cutset of size $S$ at most 2.
- If $S$ is of size 1, conclude.
- Assume $S = \{a, b\}$.
- If $ab \notin E(G)$, then add $ab$ to $G$ and prove that this does not create a $K_4$-minor.
- So now $S$ is a clique (we call that a **clique cutset**).
- Let $C_1$ be a connected component of $G \setminus S$ and $C_2 = G \setminus (S \cup C_1)$.
- For $i = 1, 2$, set $G_i = G[C_i \cup S]$ (The $G_i$ are often called **block decomposition**). By minimality of $G$, $\text{tw}(G_i) \leq 2$.
- Take a tree decomposition of $G_1$ and $G_2$ of width at most 2.
- By Helly both decompositions have bag that contains $a$ and $b$ together.
- Link those two bags to obtain a tree decomposition of $G$ of width 2.
Bounds for graphs with treewidth at most 2

**Theorem 16**

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

- The proof for \( \text{tw}(G) = 2 \) shows the role of separators with treewidth.
- One could hope for a general result of the type :

  \[
  \text{tw}(G) \leq k \iff G \in \text{Forb}_{\leq m}(K_{k+2}) \quad \text{FALSE}
  \]

- There exists graph with no \( K_5 \) minor with arbitrarily large treewidth. (As we will soon see, even planar graphs can have arbitrarily large treewidth).
**Theorem 17**

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

![Graphs for Theorem 17](image)
We have already proven these two sets of inequalities

\[
\begin{align*}
\omega(G) & \leq \chi(G) \leq tw(G) + 1 \\
\omega(G) & \leq \omega_m(G) \leq tw(G) + 1
\end{align*}
\]

where \(\omega_m(G)\) denotes the largest integer \(k\) such that \(G\) has a \(K_k\) minor.

**Conjecture (Hadwiger)**

\(\chi(G) \leq \omega_m(G)\).

- It is trivial that \(\omega_m(G) \leq 2 \iff G\) is a forest \(\Rightarrow \chi(G) \leq 2\).
- We saw \(\omega_m(G) \leq 3 \iff tw(G) \leq 2\) which implies \(\chi(G) \leq 3\) by the above inequalities.
- \(\omega_m(G) \leq 4 \Rightarrow \chi(G) \leq 4\) contains 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.
6 - Tree Decompositions and Separators
Proposition

Let \((T, W)\) be a tree decomposition of \(G\) and \(t_1 t_2\) 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 \(T \setminus t_1 t_2\) containing \(t_i\), and \(G_i\) the subgraph of \(G\) induced by \(\bigcup_{t \in T_i} (W_t \setminus S)\). Then there are no edges between \(G_1\) and \(G_2\).

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

Proposition

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 \(\text{tw}(G) = \max_{i \in I}(\text{tw}(H_i))\).

Exercice 8

Prove that the same is true for the chromatic number, and for the function which associates to a graph the size of its largest clique minor.
Clique Sums

**Definition**

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 $G_2$.

The previous proposition can be restated in terms of clique sums.

**Proposition**

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

And another characterization of treewidth

**Proposition**

$G$ has treewidth at most $k$ if and only if it can be constructed recursively by clique sum operations starting from graphs on at most $k + 1$ vertices.
This proposition will also be very useful to design algorithms.

**Proposition**

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 $|V(G)|/2$

This also permits to prove that the $2k \times 2k$-grid has treewidth at least $k$ (in fact we will prove later that its treewidth is $2k$).

**Corollary**

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

**Corollary**

The class of planar graph has unbounded treewidth.
**Exercice 9**

Prove that if $H$ is a subdivision of $G$, then $\text{tw}(H) = \text{tw}(G)$

The following exercise says that classes of graphs with bounded treewidth are sparse.

**Exercice 10**

Show that graphs $G$ of treewidth at most $k$ with $k \geq 1$ have strictly less than $k|V(G)|$ edges.

Next exercise is very important to design algorithm based on the tree decomposition.

**Exercice 11**

Show that every graph $G$ admits a tree decomposition of width $\text{tw}(G)$ with at most $|V(G)|$ bags.
Exercises on treewidth - 2

**Exercise 12**

Determine the treewidth of a path, a tree, a complete graph, a complete bipartite graph, the cube.

**Exercise 13**

Prove that if $G$ contains (as a subgraph) a complete bipartite graph with parts $A$ and $B$, then in every tree decomposition there exists a bag that contains $A$ or a bag that contains $B$.

**Exercise 14**

Prove that if $x$ and $y$ are two vertices that are joined by $k + 1$ internally vertex disjoints paths, then in every tree decomposition of $G$ of width at most $k$, there exists a bag containing both $x$ and $y$.

**Exercise 15**

Prove that if $G$ is $K_{2,3}$-minor-free then $tw(G) \leq 3$. 
7 - Duality - Cops and Robbers
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 $G$.

The proof of the previous proposition uses solely the fact that subgraphs of size at least $n/2$ form a bramble.

Note that if $G$ contains a complete $K_p$ as a minor then the connected subgraphs of $G$ associated to each vertex of $K_p$ form a bramble (no intersection, just touching).
A bramble of order 4 of $G_{3,3}$:
**Proposition**

If \((T, W)\) is a tree decomposition of \(G\) and \(\mathcal{B}\) is a bramble in \(G\), then there exists \(t \in T\) such that \(W_t\) is a transversal of \(\mathcal{B}\).

Proof by the usual "orientation of edges of the tree" argument. Therefore \(bn(G) \leq tw(G) + 1\).

The converse inequality is true but harder to prove.

It gives the following sort of minmax theorem (in fact maxmin=minmax).

**Theorem 18** (Seymour and Thomas, 1993)

For every graph \(G\), \(bn(G) = tw(G) + 1\).

**Exercice 16**

Prove that the treewidth of the grid \(G_{n,n}\) is equal to \(n\).
A Game of Cops and Robber

- 2 player game on a graph: one controls Robber, the other control Cops
- Goal of the cops is to capture the robber
- Many variants exist

In our variant:

- Cops and robbers are standing on vertices of the graph
- At each turn a fraction of the cops can move by helicopter and land on any vertex of the graph.
- The robber sees an 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 at some point they occupy all vertices adjacent to the position of the robber, and an extra cop lands by helicopter on the robber.

**Definition**

The **cop number** of a graph $G$, denoted $cn(G)$, is the smallest number of cops to ensure the capture of the robber.
**PROPOSITION**

\[ cn(G) \leq tw(G) + 1 \]

- Put every cop on 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.
- \( W_t \cap W_{t'} \) separates the component containing the robber from the rest of the graph.
- At the next move, cops in \( W_t \setminus W_{t'} \) move to occupy all of \( W_{t'} \).
- Cops apply this strategy until it reaches some leaf of the tree and the robber cannot escape.
**Proposition**

\[ \text{bn}(G) \leq \text{cn}(G) \]

- Let \( B \) be a bramble of order \( \text{bn}(G) \) and assume only \( \text{bn}(G) - 1 \) cops.
- Let \( C \) be the set of initial positions of the cops.
- By definition there exists a set \( X \in B \) such that \( X \cap C = \emptyset \).
- The robber moves to some vertex \( x \in X \).
- After that, the game really begins, cops move so that the new set occupied by the cops is \( C' \).
- Again there exists \( X' \in B \) 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,
- The robber can freely move from \( X \) to \( Y \) and this strategy can be applied for ever.
8 - Treewidth and Planar Graphs
We have seen that $tw(G_{k,k}) = k$, so if $tw(G) < k$, $G$ is $G_{k,k}$ minor free.

The following (difficult) theorem gives an approximate converse statement.

**Theorem 19** *(Excluded Grid Theorem)*

There exists $f(k)$ such that if $G$ is $G_{k,k}$-minor free then $tw(G) < f(k)$

(In 2013, Chekuri and Chuzhoy, proved the first polynomial bound)

(If $G$ is planar, one can prove $f(k) = 4k$)
Very (very) rough idea of the proof:
Let $G$ be a graph with very large treewidth. We want to show that $G$ contains a large grid.

- Show that $G$ contains a large family $\{A_1, \ldots, A_m\}$ of pairwise disjoint connected subgraphs such that:
- each pair $A_i, A_j$ can be linked in $G$ by a family $P_{i,j}$ of many disjoint $A_i - A_j$ paths avoiding the other sets.
- We then consider all the pairs $P_{i,j}, P_{i',j'}$.
- If we can find such a pair such that many of the paths in $P_{i,j}$ meets many of the path in $P_{i',j'}$, then we can find a large grid (this is the most difficult part of the proof because the intersections might be very messy).
- Otherwise, for every pair $P_{i,j}, P_{i',j'}$, many of the paths in $P_{i,j}$ avoid many of the path in $P_{i',j'}$.
- We can then select one path $P_{i,j} \in P_{i,j}$ from each family such that these selected path are pairwise disjoint.
- Contracting each of the connected subgraph will then give us a $K_m$-minor, which contains a large grid.
Planar Graphs are WQO

Tentative proof of Wagner’s Conjecture: Let \((G_n)_{n \in \mathbb{N}}\) be a sequence of graphs

- If there exists \(i > 1\) such that \(G_1\) minor of \(G_i\), WIN
- If not, consider the class \(C\) of \(G_1\)-minor free graphs.
- If we can prove that this class \(C\) has bounded treewidth, then WIN.

This is true

- if \(G_1\) is a grid by Excluded Grid Minor Theorem,
- for any \(G_1\) planar because every planar graph is the minor of some large enough grid (why? do a drawing)

This proves one direction of the following (the other one is easy, why?)

**Theorem 20**

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

**Corollary**

The class of planar graphs is wqo for the minor relation.
9 - Erdős-Posa Property
Transversals

One natural question about cycles in a graph $G$ is to determine the minimum number of vertices that one needs to delete in order to get an acyclic graph.

Equivalently this is the minimum size of a set $X$ of vertices such that $X \cap C \neq \emptyset$ for every circuit $C$ of the graph.
Such a set is often called a **transversal** of the family of circuits, and we denote here the minimum size of a transversal by $\tau_C(G)$.

Clearly if there exists in our graph $k$ pairwise vertex-disjoint circuits, then a transversal contains at least $k$ vertices.
If $\nu_C(G)$ denotes that maximum number of pairwise vertex-disjoint circuits:

$$\nu_C(G) \leq \tau_C(G)$$

A classical result or Erdős and Posa proves that for $\nu_C$ fixed, $\tau_C$ cannot be arbitrarily large.

**Theorem 21 (Erdős-Posa,’65)**

There exists a function $f : \mathbb{N} \rightarrow \mathbb{N}$, such that for any graph $G$:

$$\tau_C(G) \leq f(\nu_C(G))$$
Erdős-Posa property

- Being a transversal of the family of circuits is the same as being a transversal of the family of subgraphs of $G$ having $K_3$ as a minor (we call these subgraphs extensions of $K_3$ in $G$).

- Can we generalise Erdős and Posa’s theorem and try to ask the same question with any graph $H$ instead of $K_3$?

- A graph $H$ is said to have the **Erdős-Posa property** if there exists as in Theorem 21 a function $f$ for the family of extensions of $H$.

- **Can we characterize the graphs $H$ that have this property?**

- The results of the previous sections permits to nicely give an answer to this question

---

**Theorem 22**

A graph $H$ has the Erdős-Posa Property if and only if $H$ is planar
We prove the connected case.

Let $H$ be a connected graph. The two following statement are equivalent:

- $H$ is planar
- $\exists f : \mathbb{N} \to \mathbb{N}$ such that for any graph $G$, either $G$ contains $k$ disjoint extensions of $H$ in $G$ or by removing at most $f(k)$ vertices from $G$ we get a $H$-minor free graph

Assume first that $H$ is connected planar and let $G$ be a graph such that the maximum number of disjoint extensions of $H$ in $G$ is equal to $k - 1$.

Denote by $H_k$ the graph which is the disjoint union of $k$ copies of $H$.

So $G$ contains $H_{k-1}$ as a minor but not $H_k$.

$H_k$ being planar, by Theorem 20 there exists an integer $t_k$ such that, if a graph has treewidth at least $t_k$, then it contains $H_k$ as a minor.

So $\text{tw}(G) < t_k$.

Let us define recursively the function $f$ by $f(0) = 0$ and $f(k) = 2f(k - 1) + t_k$ for any $k \geq 1$.

We are going to prove by induction on $k$ this is the desired function so assume the result is true up to $k - 1$.

Fix an optimal tree decomposition $(T, W)$ of $G$. For an edge $t_1 t_2$ of $T$, let $T_1$ and $T_2$ be the connected components of $T \setminus t_1 t_2$ and denote for $i = 1, 2$

$$G_i := \bigcup_{t \in T_i} (W_t \setminus W_{t_i})$$
We now orient every edge \( t_1 t_2 \) towards \( t_i \) if \( G_i \) contains \( H \) as a minor. Note that doing so every edge can receive one, two or no orientation.

If every edge gets at most one direction, then it means some vertex \( t \) of the tree has no outgoing edge, which means that \( W_t \) intersects every extension of \( H \) in \( G \), and since \( |W_t| \leq t_k \leq f(k) \), we get our desired transversal.

If this is not the case, some edge \( t_1 t_2 \) gets two orientations, which means both \( G_1 \) and \( G_2 \) contain \( H \) as a minor. Since they are disjoint, if one of these two graphs contains \( k - 1 \) disjoint extensions of \( H \), then \( G \) contains \( k \) disjoint copies and we are done.

Therefore we can assume that \( G_1 \) and \( G_2 \) do not contain \( H_{k-1} \) as a minor. By induction there exists \( f(k - 1) \) vertices in each graph whose removal leaves each graph without any \( H \) minor. By taking these two sets plus \( W_{t_1} \), one gets the desired transversal of size at most \( f(k) \).
Now let us prove the converse and assume \( H \) is non planar.

We will show a construction of graphs where there are no two disjoint \( H \)-extensions (\( \nu = 1 \)), but where the number of vertices to be removed to hit every \( h \)-extension is arbitrarily large.

Let \( \Sigma \) be a surface in which \( H \) embeds, such that \( \Sigma \) is minimal genus wise.

This minimality ensures that any two drawings of \( H \) intersect.

Now one can construct a graph by embedding \( 2k+1 \) vertex disjoint copies of \( H \) in \( \Sigma \) such that no point of \( \Sigma \) is contained in more than 2 of these copies.

Transform every crossing of edges into a vertex of degree 4.

The resultant graph \( G \) has no transversal of size \( k \), but every two copies of \( H \) intersect.
10 - Graphs are WQO

**Warning : contains major handwaving**
Wagner’s Conjecture: Sketch

- Starts as before: Assume \((G_n)_{n \in \mathbb{N}}\) is a counterexample.
- We can assume that no graph \(G_i\) with \(i \geq 2\) has \(G_1\) as a minor.
- Get some structure for these graphs?
- Sufficient to get a structure theorem for all graphs not containing \(K_k\), for \(k\) fixed as a minor.
- For \(k \leq 4\) we have seen characterizations.
- For \(k = 5\), there is one due to Wagner

**Theorem 23 (Wagner - 1937)**

\(K_5\)-minor free graphs are constructed by a sequence of clique sums operations starting from \(W_8\) and planar graphs.

- For larger \(k\), Structure Theorem gives approximate characterization
Now let us define a class $G_k$ of **Almost $k$-embeddable** graphs

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** (local perturbation of a face of the embedding)

iii. Add at most $k$ **apexes** (vertices linked arbitrarily to the rest of the graph)
Theorem 24

For every graph $H$, there exists an integer $k$ such that the class of $H$-minor free graphs is obtained by a sequence of clique sum operations starting from almost $k$-embeddable graphs.

H-Minor-Free

\[ \bigcup \bigcup \bigcup \\quad \text{Bounded Genus} \quad \text{Planar} \]
"Proof" of Wagner Conjecture

Very (very) roughly, the proof that graphs are WQO for minor ordering is

- Show that graphs of bounded genus are WQO by induction on the genus (very hard).
- Almost $k$-embedable graphs are taken care to the cost of more very hard work.
- Kruskal’s Theorem’s proof is adapted to deal with the tree structure given by the clique sums operations.
Algorithmic implications

General message:

- if something works for planar graphs,
- then we might generalize it to bounded genus graphs,
- then we might generalize it to $H$-minor-free graphs.

What next?

What about topological minors?
H-topological minor free graphs look like that (Grohe and Marx, 2012)
Decomposition theorem for $H$-topological minor free graphs

**Theorem (Grohe and Marx, 2012)**

For every $H$, there is an integer $k$ such that every $H$-subdivision-free graph has a tree decomposition where the torso of every bag is either:

- $k$-almost embeddable in a surface of genus at most $k$ or
- has degree at most $k$ with the exception of at most $k$ vertices (“almost bounded degree”).

**General message:**
If a problem can be solved both

- on (almost-)embeddable graphs and
- on (almost-)bounded degree graphs,

then these results can be raised to $H$-subdivision-free graphs without too much extra effort.
11 - Rooted disjoint path problem and minor detection
In the introduction of this course we saw the following two "connectivity" problems

**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 $B$?

As we have seen, this one corresponds to a structural minmax Theorem (Menger’s theorem) in terms of separators (and hence is in co-NP) but is in fact polynomially solvable by max-flow techniques.

**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 one is NP-complete (for $k \geq 2$) but as already said, Robertson and Seymour gave a polynomial algorithm for fixed $k$, which 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. 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.

\[ k=4 \]

\[
\begin{array}{c}
\text{X} \\
\quad \\
\quad \\
\quad \\
\end{array}
\]

\[
\begin{array}{c}
\text{Y} \\
\quad \\
\quad \\
\quad \\
\end{array}
\]
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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.
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. One is the following: in some situations it is possible to prove that there exists vertex \( v \) which is \textbf{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 \textbf{big clique}.
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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.
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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**

\[ k=4 \]

2) No \( 2k \) disjoint paths

1. \( X \)
2. \( Y \)
3. \( S \)
4. \( C \)

clique of size \( 2k \)
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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**
Idea : Irrelevant Vertices

- Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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

\[
\begin{align*}
\text{k=4} & \quad 2) \text{ No } 2k \text{ disjoint paths} \\
X & \quad s = |S| \text{ disjoint paths} \\
\text{Y} & \quad \text{Irrelevant??} \\
S & \quad \text{clique of size } 2k \\
C &
\end{align*}
\]
Idea: Irrelevant Vertices

- Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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

$$k=4$$

2) No $2k$ disjoint paths

Suppose there exists $k$ disjoint rooted paths from $X$ to $Y$
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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 there exists $k$ disjoint rooted paths from $X$ to $Y$
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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 there exists \( k \) disjoint rooted paths from \( X \) to \( Y \)

Suppose there exists \( k \) disjoint rooted paths from \( X \) to \( Y \)

Irrelevant??

clique of size \( 2k \)
Let us describe rapidly the ideas behind the algorithm for rooted disjoint path problem. 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

\[ k=4 \]

2) No $2k$ disjoint paths

Suppose there exists $k$ disjoint rooted paths from $X$ to $Y$

$\rightarrow$ Then there exists $k$ disjoint rooted paths from $X$ to $Y$ avoiding the irrelevant
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 19 we know that if the treewidth is large enough, such a big grid minor exists.

So if no big grid minor, there we are in the case of bounded treewidth, and we will see in the next chapter that dynamic programming approaches permits us to solve this kind of problem.

If big grid minor, there exists irrelevant vertices but can we efficiently find them? Two cases:

- first if the graph contains a large clique minor (difficult but not too difficult)
- then if no large clique minor exists, Robertson and Seymour use their structure Theorem 24
12 - Computing Tree Decompositions
Soon we will show how to take advantage of a fact that input graphs have **bounded treewidth** in order to solve hard problems.

We will show that some classical NP-Hard problems (like max clique, min Vertex Cover...) have a **FPT algorithm** when parametrized by tree width complexity: \( O(f(tw(G))n^{O(1)}) \)

But before that we need to know whether we can **efficiently compute a low treewidth decomposition** for a graph whose treewidth is small ??
Computing tree width

The first "trouble" :

**Problem** (Computing tree width)

**Input** : $G$, $w$

**Output** : TRUE if $\text{tw}(G) \leq w$

is NP-Hard : Arnborg, Corneil, Proskurowski ’87 (note that polytime open for planar)

**But there exists FPT** : $O(2^{w^2} n)$ algorithm (Bodlaender 96)

There even exists approximations algorithms with FPT complexity (and even polynomial)

**Theorem 25**

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

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

This is enough for our FPT algorithms seen before : simply run this 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 $O(f(\text{tw}(G)).n^2)$.

Pierre Charbit - charbit@irif.fr

Minors - Treewidth - Algorithms MPRI Graph Algorithms
DEFINITION

A set $S$ is a **good separator** for a set $W$ of vertices 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).

We have already proven a statement regarding these:

**Proposition**

If $\text{tw}(G) < k$, every $X$ of size at least $2k + 1$ admits a good separator of size at most $k$.

The main ingredient for the proof is the fact that the converse is almost true.

**Theorem 26**

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 $\text{tw}(G) \leq 4k - 1$. 
We prove inductively that there exists an algorithm for the following ($W = \emptyset$ to get the previous theorem).

**Problem**

**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 $tw(G) > k$.
- So $W$ has a good separator. Assume for the moment that it exists and we are able to compute it.
• S is a good separator for W
• G \ 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
• Apply induction on G'_i = G_i \cup S, i = 1, 2
• Either certificate that \( tw(G_i) > k \) for some i and therefore \( tw(G) > k \)
• Or get two rooted decompositions T_1, T_2 of G'_1 and G'_2 with W_i \subset root(T_i)
• Add a root bag containing all vertices in W \cup S (note that W \cup S − 1 \leq 4k − 1) attached to the roots of T_1 and T_2.
Algorithm to find good separator $S$?

$S$ 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 $2|W|/3$,
- $W_0$ is a subset of a separator of size at most $k$ separating $V_1$ and $V_2$ where $W_i \subseteq V_i$

Equivalently if and only if in $G \setminus W_0$, there are at most $k - |W_0|$ disjoint paths from $W_1$ to $W_2$.

Ford Fulkerson : $O\left(k^2 n\right)$ (because the graph has at most $kn$ edges, otherwise it cannot have treewidth at most $k$)

$3^{3k}$ ways of defining the partition $W_0$, $W_1$, $W_2$ so $O\left(27^k k^2 n\right)$ for this step

therefore complexity $O\left(27^k k^2 n^2\right)$ in total since the tree decomposition has at most $n$ nodes.
13 - Algorithms for bounded-treewidth graphs
A **parametrized algorithmic** problem is a problem where a certain parameter $k$ is given in addition to the input (of size $n$).

- $k$ can be the size of the solution, or an implicit parameter of the input graph (diameter, maximum degree, treewidth ..).
- The complexity is studied as a function of $n$ and $k$.

Let us distinguish three possibilities for a parametrized algorithmic problem.

- **Either the problem is already hard for fixed $k$.**
  
  Example = decide if $\chi(G) \leq k$ is NP hard for $k = 3$.

- **Or the problem is NP hard with $k$ in the input but polynomial for $k$ fixed.**
  
  Example : Decide if $\alpha(G) \leq k$ with parameter $k$
  
  Solvable by exhaustive search : $O(n^k)$.

- **FPT : Algorithm in time** $O(f(k)n^{O(1)})$

We will now show that **many NP hard problems are FPT when parametrized by treewidth**.
Here is a weighted version of the classical problem of finding the maximum size of an independent set.

**Problem (Maximum Weighted Independent Set - MWIS)**

**Input**: Graph with weight function \( w : V(G) \rightarrow \mathbb{R} \)

**Output**: an Independent set of maximum weight.

- NP-complete for general graphs.
- We will first show why it is easy and polytime solvable on trees (by dynamical programming), before extending this to graphs of bounded treewidth.
MWIS for Trees

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:

- $W(v)$ 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 $v$

The value of maximum weight independent set of $T$ is precisely $W(r)$.
We can compute inductively $W$, $W^+$, and $W^-$, since

$$W(v) = \max(W^+(v), W^-(v))$$
$$W^+(v) = w(v) + \sum_{u \in C(v)} W^-(u)$$
$$W^-(v) = \sum_{u \in C(v)} W(u)$$
Theorem 27

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

Proof:

- Root the tree-decomposition: denote by $G_t$ the subgraph induced by vertices that belong to bags $W_{t'}$ where $t'$ is a descendant of $t$.
- Bottom-up dynamical programming
- For every node $t$ of the tree and every subset $S$ of $W_t$ one need to store the value
  $$\phi_t(S) = \max\{w(S'), S' \text{ independent set of } G_t \text{ such that } S' \cap W_t = S\}.$$
- Easy to compute for leaf
- For a node $t$ with children $t_i$: by exhaustive search find $S_i \subset W_{t_i}$ such that:
  - $S_i$ independent set
  - $S \cap W_{t_i} = S_i \cap W_t$
  - $\phi(S_i, t_i)$ maximal.

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

To design algorithms parametrized by treewidth, it is convenient to use the following particular tree decompositions.

**Definition**

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:

- **Leaf**: $t$ has no child and $|W_t| = 1$.
- **Join**: $t$ has two children $t_1$ and $t_2$ and $W_t = W_{t_1} = W_{t_2}$.
- **Introduce**: $t$ has one child $t'$ and $W_t = W_{t'} \setminus \{x\}$ where $x \notin W_{t'}$.
- **Forget**: $t$ has one child $t'$ and $W_t = W_{t'} \cup \{x\}$ where $x \notin W_t$.
From tree decomposition to nice tree decomposition

**Theorem 28**

A tree decomposition of width $k$ and $n$ nodes can be turned into a nice tree decomposition of width $k$ and $O(w \cdot n)$ nodes in time $O(w^2 \cdot n)$.

**Proof Sketch** :
Root the decomposition arbitrarily.
For each internal node with $p$ children, it is possible to add $2p$ new join nodes to make it binary.
For each edge $t_1 t_2$ replace $t_1 t_2$ by a path with at most $k$ forget nodes and at most $k$ introduce nodes. □

Using nice decomposition, it becomes super easy to compute $M[t, S]$ in a bottom-up fashion as we are going to see now.
MWIS with nice decompositions

For each node $t$ and each subset $S$ of $W_t$:

$$M[t, S] = \max \text{ weighted independent } I \text{ such that } I \subseteq V_t \text{ and } I \cap W_t = S.$$ 

- **Leaf** : $|W_t| = 1$, trivial
- **Introduce** : one child $t'$ with $W_t = W_{t'} \cup v$:

$$M[t, S] = M[t', S]$$

$$= M[t', S \setminus \{s\}] + \omega(v)$$

$$= -\infty$$

if $v \notin S$

if $v \in S$

if $S$ contains $v$ and a neighbor of $v$

- **Forget** : one child $t'$ with $W_t = W_{t'} \setminus v$:

$$M[t, S] = \max(M[t', S], M[t', S \cup \{v\}])$$

- **Join** : $t$ has two children $t_1$ and $t_2$ such that $W_t = W_{t_1} = W_{t_2}$:

$$M[t, S] = M[t_1, S] + M[t_2, S] - \omega(S)$$

Pierre Charbit - charbit@irif.fr
Hamiltonian Circuit

Let us use nice decompositions to solve Hamiltonian Circuit.

We have a nice tree decomposition \((T, W)\). For a node \(t\) of \(T\) we denote as usual \(V_t\) the set of vertices of \(G\) that appear in the bags \(W_t\) where \(t'\) is a descendant of \(t\).

For each partition of \(W_t\) into \((W_t^0, W_t^1, W_t^2)\) and each matching \(M\) of the vertices of \(W_t^1\), we want to know if there exists (and if yes we store one solution) a set of paths in \(V_t\) such that =

- the paths cover all vertices in \(V_t\) except the vertices in \(W_t^0\)
- the end points of the paths are exactly the vertices in \(W_t^1\)
- For every edge in \(M\), its two ends are the two endpoints of the same path.
- The vertices in \(W_t^2\) all belong to the same path

Note that there are at most \(3^{\text{tw}(G) \cdot \text{tw}(G)}\) such things to consider.

Of course solving the problem for the root bag \(W_r\) with \(W_r^0 = W_r^1 = \emptyset\) will find an hamiltonian cycle of the graph.
Hamiltonian Circuit

Bag $W_t$
Hamiltonian Circuit

Bag $W_t$

$W_t^0$ $W_t^1$ $W_t^2$

- - - - - - - - - -

- - - - - - - - - -
Hamiltonian Circuit

Bag $W_t$

$W_t^0$  $W_t^1$  $W_t^2$
Case Forget Node $v$

Bag $W_t$

$W_t^0$  $W_t^1$  $W_t^2$
Case Forget Node $v$

**Bag $W_t$**

- $W_t^0$
- $W_t^1$
- $W_t^2$

**Bag $W'_t = W_t \cup \{v\}$**

- $W_t^0$
- $W_t^1$
- $W_t^2$

Problem Equivalent to ($W_t^0$, $W_t^1$, $W_t^2 \cup \{v\}$) for node $t'$
Case Introduce Node

Subcase 1 = $v \in W_0$

Bag $W_t$

$W_0^t$ $W_1^t$ $W_2^t$

$W_t' = W_t \setminus \{v\}$

$W_0^t$ $W_1^t$ $W_2^t$
Case Introduce Node
Subcase 1 = $v \in W_t^0$

Bag $W_t$

$W_t^0$

$W_t^1$

$W_t^2$

Bag $W_t' = W_t \setminus \{v\}$

$W_t^0$

$W_t^1$

$W_t^2$
Case Introduce Node

Subcase 1 = $v \in W_t^0$

Problem Equivalent to $(W_t^0, W_t^1, W_t^2 \cup \{v\})$ for node $t'$
Case Introduce Node

Subcase 2 = \( v \in W_t^1 \)

Bag \( W_t \)

\[ W_t^0 \quad W_t^1 \quad W_t^2 \]

\[ \bullet \quad \bullet \quad \bullet \quad v \quad \bullet \quad \bullet \]

Bag \( W_t' = W_t \setminus \{v\} \)

\[ W_t^0 \quad W_t^1 \quad W_t^2 \]

\[ \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \]
Case Introduce Node

Subcase 2 = $v \in W_t^1$

Bag $W_t$

$$Bag \ W_t' = W_t \setminus \{v\}$$

$W_t^0$ $W_t^1$ $W_t^2$

$v$

$W_t^0$ $W_t^1$ $W_t^2$
Case Introduce Node

Subcase 2 = \( v \in W_t^1 \)

Bag \( W_t \)

\[
W_t^0 \quad W_t^1 \quad W_t^2
\]

Bag \( W'_t = W_t \setminus \{v\} \)

\[
W'_t^0 \quad W'_t^1 \quad W'_t^2
\]

Every neighbour of \( v \) in \( V_t \) is in \( W_t \)

Pierre Charbit - charbit@irif.fr

Minors - Treewidth - Algorithms MPRI Graph Algorithms
Case Introduce Node

Subcase 2 = \( v \in W_t^1 \)

**Bag** \( W_t \)

\[
W_t^0 \quad W_t^1 \quad W_t^2
\]

Every neighbour of \( v \) in \( V_t \) is in \( W_t \)

Bag \( W'_t = W_t \setminus \{v\} \)

**Bag** \( W'_{t1} \quad W'_{t2} \quad W'_{t3} \)

\( v \) \quad \( w \)
Case Introduce Node

Subcase 2 = \( v \in W_t^1 \)

Bag \( W_t \)

\[
\begin{align*}
W_t^0 & \quad W_t^1 & \quad W_t^2 \\
& \quad v & \quad w \\
\end{align*}
\]

Bag \( W'_t = W_t \setminus \{v\} \)

\[
\begin{align*}
W_t^0 & \quad W_t^1 & \quad W_t^2 \\
& \quad w \\
\end{align*}
\]

Every neighbour of \( v \) in \( V_t \) is in \( W_t \)

Problem equivalent to the \( t' \) problem \((W_t^0, W_t^1 \setminus \{v\} \cup \{w\}, W_t^2 \setminus \{w\})\) for some vertex \( w \in W_t^2 \)
Case Introduce Node

Subcase 3: $v \in W_{t_1}^1$

Bag $W_t$

$W_{t_0}^0 \quad W_{t_1}^1 \quad W_{t_2}^2$

Bag $W'_t = W_t \setminus \{v\}$

$W_{t_0}^0 \quad W_{t_1}^1 \quad W_{t_2}^2$
Case Introduce Node

Subcase 3 = $v \in W_t^1$

Bag $W_t$

![Diagram of Bag $W_t$ with $W_t^0$, $W_t^1$, and $W_t^2$]

Every neighbour of $v$ in $V_t$ is in $W_t$

Bag $W_t' = W_t \setminus \{v\}$

![Diagram of Bag $W_t'$]

Problem equivalent to some $t'$ problem
Case Introduce Node

Subcase 3 = \( v \in W^1_t \)

\[ \text{Bag } W_t \]

\[ W^0_t \quad W^1_t \quad W^2_t \]

\[ W^0_t = W_t \setminus \{v\} \]

Every neighbour of \( v \) in \( V_t \) is in \( W_t \)

Problem equivalent to some \( t' \) problem
Case Introduce Node

Subcase 3 = \( v \in W^1_t \)

Bag \( W_t \)

\[
W^0_t \quad W^1_t \quad W^2_t
\]

Every neighbour of \( v \) in \( V_t \) is in \( W_t \)

Bag \( W'_t = W_t \setminus \{v\} \)

Problem equivalent to some \( t' \) problem
Case Introduce Node

Subcase 3 = \( v \in W_t^1 \)

Every neighbour of \( v \) in \( V_t \) is in \( W_t \)

Problem equivalent to some \( t' \) problem
Case Introduce Node

Subcase 3 = $v \in W_t^1$

Bag $W_t$

$W_t^0$ $W_t^1$ $W_t^2$

Bag $W_t' = W_t \setminus \{v\}$

$W_t^0$ $W_t^1$ $W_t^2$

Every neighbour of $v$ in $V_t$ is in $W_t$

Problem equivalent to some $t'$ problem
Case Merge

Bag $W_t = W_{t1} = W_{t2}$

Easy to use the info for $t_1$ and $t_2$ to decide if a solution exists for each problem of $t$
Other Problems

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

**Theorem 29**

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 $O(2^k.k^k.n)$
3. Computing $\gamma(G) := \min\{|X|, X \text{ dominating set }\}$, can be done in time $O(f(k).n)$.  

Pierre Charbit - charbit@irif.fr
14 - Courcelle Theorem
A celebrated algorithmic meta-theorem of Courcelle generalises all the previous results to monadic second order formulas.

Logical formulas are constructed inductively using

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

First order formulas: quantifiers over vertices and edges (like \( \forall v \exists e \ldots \)),
- \( \text{MSO}_1 = \text{FO} + \text{quantify over sets of vertices} \),
- \( \text{MSO}_2 = \text{MSO}_1 + \text{quantify over sets of edges} \).
This is a second order formula for 3 colourability:

\[ \exists X_1 \subset V \exists X_2 \subset V \exists X_3 \subset V \]

\[ (\forall x \in V) (x \in X_1 \lor x \in X_2 \lor x \in X_3) \]

\[ \land \neg(x \in X_1 \land x \in X_2) \land \neg(x \in X_1 \land x \in X_3) \land \neg(x \in X_2 \land x \in X_3) \]

\[ \land (\forall xy \in E) \neg(x \in X_1 \land y \in X_1) \land \neg(x \in X_2 \land y \in X_2) \land \neg(x \in X_3 \land y \in X_3) \]
Theorem of Courcelle asserts that every such property is easy to decide for bounded treewidth graphs.

**Theorem 30**

Let $\phi$ be a MSO$_2$ formula of length $k$, and let $G$ be a graph on $n$ vertices with $\text{tw}(G) \leq t$. There exists an algorithm that performs in time $f(k, t).O(n)$ to decide whether $G$ satisfies $\phi$.

**Exercice 17**

Prove that the existence of a hamiltonian cycle can be expressed as an MSO$_2$ formula. Same for the fact that a fixed graph $H$ is a minor of the input graph.
15 - FPT parametrized by size of the solution
TreeWidth and Other Techniques
We want here FPT algorithms *parametrized by size of solution*.

Example:

<table>
<thead>
<tr>
<th><strong>Problem</strong> (Vertex Cover)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong> : $G, k$</td>
</tr>
<tr>
<td><strong>Output</strong> : Does there exist a set $X$ of size at most $k$ such that every edge is incident to at least one vertex of $S$?</td>
</tr>
</tbody>
</table>

- Vertex Cover = Complement of Independent Set.
- Therefore it is **NP-Hard**
- Also, without parameter, or parametized by treewidth, this is equivalent to the **Maximum Independent Set** Problem.
- **BUT** parametrized by size of solution the problems are not similar: having a FPT for one does not imply the same for the other.
- In fact no $n^{o(k)}$ algorithm is known to decide if max independent set $\geq k$, whereas there are many FPT algorithms for $k$-Vertex Cover (as we will see).
Existence of FPT via Robertson and Seymour Theorems

Here we show a general technique to prove the existence of an FPT algorithms for the class of problems described below. $C$ denotes a class of graphs.

**Problem** $(C + k$ vertices$)$

**Input:** $G, k$

**Output:** Does there exist a set $X$ of size at most $k$ such that $G \setminus X \in C$?

It generalizes several well known problems:

- $C =$ stable sets $\Rightarrow k$-Vertex Cover
- $C =$ forests $\Rightarrow k$-Feedback Vertex Set Problem

We can answer this problem if $C$ is **minor-closed**:

- if so, the set of YES instances is also minor closed (why?)
- $\Rightarrow f(k)O(n^3)$ algo by Robertson and Seymour theorems.
- **Problems**: this is just existential proof since we do not know how to get the bounds (Robertson and Seymour theorems are not constructive). Moreover not uniform: not the same algorithm for different values of $k$ (forbidden minors depend on $k$).
FPT algorithms through Tree Width methods

For some problems, an FPT parametrized by treewidth implies an FPT parametrized by the size of the solution.

**Example :** $k$-Vertex Cover.

- Decide whether $\text{tw}(G) \leq k$ and find a tree decomposition of width $k$ in that case
- if $\text{tw} > k$ answer NO (why ?)
- if $\text{tw} \leq k$ and with tree dec, decide $G$ has a $k$-VC with Dynamic Programming

Both algorithmic steps have FPT algos.

This technique will work as long as the problem we are solving is such that there is a bound $t$ such that if $\text{tw}(G) > t$ then either the answer is always YES or always NO.
This concerns Planar Graphs and many problems.

**Definition** (*k*-Outerplanar Graphs)

- A planar graph is said to be **outerplanar** (or 1-outerplanar) if there exists a planar embedding of $G$ where all its vertices lie on the outer face (for example, $K_4$ is planar but not outerplanar).
- A embedding of a graph is **$k$-outerplanar** if by deleting its outer face, one gets an embedding that is $k - 1$-outerplanar graph.
- A graph is **$k$-outerplanar** if has a $k$-outerplanar embedding.

**Proposition**

$k$-outerplanar graphs have treewidth at most $3k$.

Proof:

- triangulate (only makes the tw bigger) and add a vertex $r$ connected to the outerface. $\Rightarrow G'$
- There is a spanning tree $T$ rooted at $r$ of depth at most $k$
- Let $T^*$ be the graph which vertices are the faces of $G'$ and edges linking faces if they share an edge not in $T$. $T^*$ is a forest.
- Construct a treedec of $G'$ based on $T^*$ with bag containing the vertices of the face $f$ and al of its ancestors in $T$. 
Let $G$ be a planar graph with a planar embedding, let $L_1$ be its outer face and for $i \geq 1$, $L_i$ be the outer face of $G \setminus \bigcup_{j<i} L_j$.

For $0 \leq s \leq k$, let $G_s$ obtained by deleting all layers $L_i$ with $i \equiv s \mod [k + 1]$.

$G_s$ is $k$-outerplanar so one can apply bounded treewidth algo for each $s$.

For at least one value $s$, one does not delete any of the $k$ vertices of the solution.

Works for Subgraph isomorphism, Independent Set, VertexCover, Dominating Set ...

Works also to get Polynomial Time Approx Scheme (PTAS) for the same problems.
Another powerful technique for planar graphs. Again We illustrate with $k$-Vertex Cover. 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 (planar excl. Grid Minor Theorem)
- 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 $2^{O(\sqrt{k})} \cdot n^c$:

- If $tw > 4 \sqrt{2k + 1}$ one can answer NO,
- otherwise we do dynamic programming as seen before

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.
We prove an easy $2^k \cdot O(n)$ algorithm for $k$-VC by a technique called branching.

Let $xy$ be an edge, the main idea is:

| $G$ has a $k$-VC | $G \setminus x$ has a $(k - 1)$-VC or $G \setminus y$ has a $(k - 1)$-VC |

Hence, to solve an instance $(G, n, k)$ it suffices to solve two instances $(G, n - 1, k - 1)$. 
The idea of kernelization is to do the preprocessing which transforms an instance \((G, k)\) in an instance \((G', k')\), called a kernel, such that

- \((G, k)\) is a YES-instance if and only if \((G', k')\) is a YES-instance
- \(k' \leq k\)
- the size \(G'\) is bounded by a function of \(k\).

It is then sufficient to apply a brute force algorithm on \((G', k')\), to get an FPT algorithm.
We are going to construct a polynomial kernel for $k$-VC problem.

Set $G = G'$ and $k = k'$ and repeat these two steps as much as you can:

- If $v$ is an isolated vertex, it is not in the solution, set $G' := G' \setminus v$
- If $d(v) \geq k + 1$, it must be in any solution (if one exist) set $G' := G' \setminus v$ and $k' := k' - 1$

*What can you say about the obtained $G'$ ?*

If $|E(G')| > k^2$ answer **NO**

Otherwise $(G', k')$ is a kernel of size at most $2k^2$, we brute force on it

**Complexity:** $O(k^{2k} + n + m)$

By using more elaborated techniques, one can find better kernels and thus better algorithms ($2k - \text{clog}(k)$ is the best known kernelization).
Iterative Compression

This technique is a very powerful one, many of the difficult FPT algorithms are based on it. Here we continue to illustrate it via the example of **Vertex Cover** (even though in this case the technique does not yield a better algorithm than the other principles exposed before).

Central idea: instead of finding directly in FPT time a solution of type \( k \), we prove a compression result, i.e. **an algorithm that given a solution \( S \) of size \((k + 1)\) will either find a solution of size \( k \), or certify that non exist.**

Some preliminary remarks and a compression technique:

- Assume \( S \) is a solution of size \((k + 1)\)

- For every \( S' \subset S \), let \( S'' = N(S \setminus S') \setminus S \).

- If \( S \setminus S' \) contains no edge, \( T = S' \cup S'' \) is a vertex cover.

- If for some \( S' \), \( T \) has size at most \( k \) then a \( k \)-vertex cover exists, otherwise no, since any \( k \)-VC must be of this kind.
Here is now the FPT algorithm based on this observations:
If \( V(G) = \{v_1, ..., v_n\} \), and \( G_i = G[v_1, ..., v_i] \) for all \( i \in \{1, ..., n\} \), define recursively sets \( S_i \) that are Vertex Cover of \( G_i \).

- \( S_1 = \emptyset \)
- For \( i \) from 1 to \( n \) do
  - If \( S_i \) is a VC for \( G_{i+1} \) then define \( S_{i+1} := S_i \)
  - If not define \( S_{i+1} = S_i \cup \{v_{i+1}\} \)
  - If \(|S_{i+1}| > k\) use the previous compression algorithm to reduce its size by 1 or output NO and break.

Either we output NO at some point or we construct \( S_n \) which is a VC of size at most \( k \) of \( G \). This gives a \( O(2^k n) \) algorithm.
Iterative Compression

This technique will work for any minimisation problem for which we are able to construct a sequence $G_1, G_2, ..., G_n = G$ and

- a $k$-solution for $G_1$ exists.
- a $k$-solution for $G_i$ can be extended (in poly time) to a $(k + 1)$-solution of $G_{i+1}$
- if $G_{i+1}$ has a $k$ solution then $G_i$ has one
- if a $(k + 1)$-solution of $G_{i+1}$ is given, there is an FPT algorithm to decide if it has a $k$-solution.
Method proposed in 1994 by Alon, Yuster, and Zwick (see [1]).

We switch from $k$-VC to the problem of deciding whether a graph contains a path on $k$-vertices, $k$ being the parameter.

Of course this is NP-complete since $k = n$ solves hamiltonian path.

The surprising idea is to transform the problem into the following: assume every vertex is randomly given a color from $\{1, 2, \ldots, k\}$.

Now the problem is to decide in FPT time whether $G$ contains a path on $k$ vertices using all $k$-colours.

Assume first we can do that. Now if a $k$-path $P$ exists, what is the probability that it receives all $k$ colours?

The answer is $k!/k^k$. Using Stirling's formula ($k! \sim \sqrt{2\pi k} \left( \frac{k}{e} \right)^k$) we get that the probability is at least $1/e^k$.

If we repeat the argument $N$ times, the probability that $P$ is never colorful is $(1 - e^{-k})^N$ which converges to 0.

Two questions to solve:

- How to solve the coloured version?
- How to derandomize?
To solve the coloured version, first note that we can slightly transform it into a rooted version: we want the path to start in a specified vertex $s$ (it is enough to add a universal vertex with a new colour).

Define a function $f(x, C)$ which is equal to 1 if there exists a path from $s$ to vertex $x$ using all colours in set $C$ and 0 otherwise.

By dynamical programming one can compute the function: start from $x = s$ $C = col(s)$ and then go in BFS fashion. It takes time $2^k m$.

To derandomize, one can use the notion of \textit{k-perfect family of hash functions}: family of functions from a set $V$ to $\{1, \ldots, k\}$ such that for any $K \subset V$ of size $k$, there exists one function in the family such that $f$ is bijective when restricted to $K$.

Several constructions exist, in [1] they prove the existence of a family of $k$-perfect has functions for a set of size $n$, of cardinality $2^{O(k)} \log(n)$, which can be computed in time $2^{O(k)} n \log(n)$.
   Color-coding.

   \textit{Graph Theory}, volume 244.

   \textit{Graph Theory (Graduate Texts in Mathematics)}.

   Ordering by divisibility in abstract algebras.

   Sur le problème des courbes gauches en topologie.

   Zur allgemeinen kurventheorie.

   Graph minors. VII. Disjoint paths on a surface.

   Graph minors. XIII. The disjoint paths problem.
Graph minors. XX. Wagner’s conjecture.