1 - Treewidth, Forbidden Minor and planar graphs

How to prove Wagner Conjecture

Wagner Conjecture: in every infinite sequence of graphs $(G_1, G_2, \dots)$, one is the minor of another.

How to prove Wagner Conjecture? Well, the natural way is:

- Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.

Hence, understanding graphs in $Forb_{\preccurlyeq_m}(H)$ for any fixed graph H would help a lot.

For example, if for some i, $Forb_{\preccurlyeq_m}(G_i)$ has bounded treewidth, then we may assume that for every $j \geq i$, G_j has bounded treewidth and since we know that bounded tree width graphs satisfy Wagner Conjecture, we are done.

How to prove Wagner Conjecture

Wagner Conjecture: in every infinite sequence of graphs $(G_1, G_2, \dots)$, one is the minor of another.

How to prove Wagner Conjecture? Well, the natural way is:

- Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.
- We can assume that no graph G_i with $i \geq 1$ has G_0 as a minor.

Hence, understanding graphs in $Forb_{\preccurlyeq_m}(H)$ for any fixed graph H would help a lot.

For example, if for some i, $Forb_{\preccurlyeq_m}(G_i)$ has bounded treewidth, then we may assume that for every $j \geq i$, G_j has bounded treewidth and since we know that bounded tree width graphs satisfy Wagner Conjecture, we are done.

How to prove Wagner Conjecture

Wagner Conjecture: in every infinite sequence of graphs $(G_1, G_2, \dots)$, one is the minor of another.

How to prove Wagner Conjecture? Well, the natural way is:

- Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.
- We can assume that no graph G_i with $i \geq 1$ has G_0 as a minor.
- Hence $G_i \in Forb_{\preccurlyeq_m}(G_0)$ for $i \geq 1$.

Hence, understanding graphs in $Forb_{\preccurlyeq_m}(H)$ for any fixed graph H would help a lot.

For example, if for some i, $Forb_{\preccurlyeq_m}(G_i)$ has bounded treewidth, then we may assume that for every $j \geq i$, G_j has bounded treewidth and since we know that bounded tree width graphs satisfy Wagner Conjecture, we are done.

How to prove Wagner Conjecture

Wagner Conjecture: in every infinite sequence of graphs $(G_1, G_2, \dots)$, one is the minor of another.

How to prove Wagner Conjecture? Well, the natural way is:

- Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.
- We can assume that no graph G_i with $i \geq 1$ has G_0 as a minor.
- Hence $G_i \in Forb_{\preccurlyeq_m}(G_0)$ for $i \geq 1$.
- More generally, we may assume that for every $i < j$, $G_j \in Forb_{\preccurlyeq_m} G_i)$.

Hence, understanding graphs in $Forb_{\preccurlyeq_m}(H)$ for any fixed graph H would help a lot.

For example, if for some i, $Forb_{\preccurlyeq_m}(G_i)$ has bounded treewidth, then we may assume that for every $j \geq i$, G_j has bounded treewidth and since we know that bounded tree width graphs satisfy Wagner Conjecture, we are done.

How to prove Wagner Conjecture

Wagner Conjecture: in every infinite sequence of graphs $(G_1, G_2, \dots)$, one is the minor of another.

How to prove Wagner Conjecture? Well, the natural way is:

- Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.
- We can assume that no graph G_i with $i \geq 1$ has G_0 as a minor.
- Hence $G_i \in Forb_{\preccurlyeq_m}(G_0)$ for $i \geq 1$.
- More generally, we may assume that for every $i < j$, $G_j \in Forb_{\preccurlyeq_m} G_i)$.

Hence, understanding graphs in $Forb_{\preccurlyeq_m}(H)$ for any fixed graph H would help a lot.

For example, if for some i, $Forb_{\preccurlyeq_m}(G_i)$ has bounded treewidth, then we may assume that for every $j \geq i$, G_j has bounded treewidth and since we know that bounded tree width graphs satisfy Wagner Conjecture, we are done.

How to prove Wagner Conjecture

Wagner Conjecture: in every infinite sequence of graphs $(G_1, G_2, \dots)$, one is the minor of another.

How to prove Wagner Conjecture? Well, the natural way is:

- Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.
- We can assume that no graph G_i with $i \geq 1$ has G_0 as a minor.
- Hence $G_i \in Forb_{\preccurlyeq_m}(G_0)$ for $i \geq 1$.
- More generally, we may assume that for every $i < j$, $G_j \in Forb_{\preccurlyeq_m} G_i)$.

Hence, understanding graphs in $Forb_{\preccurlyeq_m}(H)$ for any fixed graph H would help a lot.

For example, if for some i, $Forb_{\preccurlyeq_m}(G_i)$ has bounded treewidth, then we may assume that for every $j \geq i$, G_j has bounded treewidth and since we know that bounded tree width graphs satisfy Wagner Conjecture, we are done.

Moreover, since $Forb_{\preccurlyeq_m}(H) \subseteq Forb_{\preccurlyeq_m}(K_{|V(H)|})$, it is enough to understand K_t-minor-free graphs.

Treewidth of minor closed class

Recall that, if H is a graph, $Forb_{\preccurlyeq_m}(H) = \{G : H \textbf{ is not a minor of } G\}$
We have already seen that:

- Graphs in $Forb_{\preccurlyeq_m}(K_3)$ have treewidth at most 1.
- Graphs in $Forb_{\preccurlyeq_m}(K_4)$ have treewidth at most 2.
- Graphs in $Forb_{\preccurlyeq_m}(K_5)$ have unbounded treewidth (because of grids).

Treewidth of minor closed class

Recall that, if H is a graph, $Forb_{\preccurlyeq_m}(H) = \{G : H \textbf{ is not a minor of } G\}$
We have already seen that:

- Graphs in $Forb_{\preccurlyeq_m}(K_3)$ have treewidth at most 1.
- Graphs in $Forb_{\preccurlyeq_m}(K_4)$ have treewidth at most 2.
- Graphs in $Forb_{\preccurlyeq_m}(K_5)$ have unbounded treewidth (because of grids).

A natural question to ask is then: **for which H, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth?** i.e. there exists a number t such that for every $G \in Forb_{\preccurlyeq_m}(H)$, $tw(G) \leq t$.

Treewidth of minor closed class

Recall that, if H is a graph, $Forb_{\preccurlyeq_m}(H) = \{G : H \textbf{ is not a minor of } G\}$
We have already seen that:

- Graphs in $Forb_{\preccurlyeq_m}(K_3)$ have treewidth at most 1.
- Graphs in $Forb_{\preccurlyeq_m}(K_4)$ have treewidth at most 2.
- Graphs in $Forb_{\preccurlyeq_m}(K_5)$ have unbounded treewidth (because of grids).

A natural question to ask is then: **for which H, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth?** i.e. there exists a number t such that for every $G \in Forb_{\preccurlyeq_m}(H)$, $tw(G) \leq t$.

One of the most important result of graph minor theory is a complete and beautiful characterization of such H.

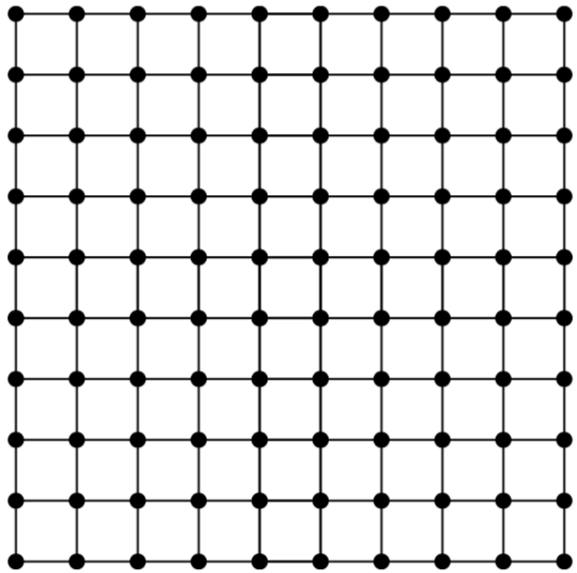
Question: For which H, graphs in $Forb_{\preccurlyeq_m}(H)$ has bounded treewidth?

First, **H must be planar.**

Question: For which H, graphs in $Forb_{\preccurlyeq_m}(H)$ has bounded treewidth?

First, **H must be planar.**

- Indeed, all grids and their minor are planar (why?).
- And grids can have arbitrarily large treewidth.
- Hence, if H is non-planar, then $Forb_{\preccurlyeq_m}(H)$ contains all grids, and thus $Forb_{\preccurlyeq_m}(H)$ does not have bounded treewidth.



Grid Minor Theorem

Theorem (Grid Minor Theorem, Robertson and Seymour, V)

Given a graph H, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth if and only if H is planar.

Grid Minor Theorem

Theorem (Grid Minor Theorem, Robertson and Seymour, V)

Given a graph H, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth if and only if H is planar.

We need to prove the *if part*, that is, for H a planar graph, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth.

Observe that every planar graph is a minor of some grid. To see this, draw a planar graph and superimpose a sufficiently fine grid, then fatten vertices and edges of the planar graph.

We denote by $G_{k,k}$ the $k \times k$ grid.

Hence, for every planar graph H, there exists k_H such that $H \preccurlyeq_m G_{k_H,k_H}$, and thus $Forb_{\preccurlyeq_m}(H) \subseteq Forb_{\preccurlyeq_m}(G_{k_H,k_H})$.

Grid Minor Theorem

Theorem (Grid Minor Theorem, Robertson and Seymour, V)

Given a graph H, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth if and only if H is planar.

We need to prove the *if part*, that is, for H a planar graph, graphs in $Forb_{\preccurlyeq_m}(H)$ have bounded treewidth.

Observe that every planar graph is a minor of some grid. To see this, draw a planar graph and superimpose a sufficiently fine grid, then fatten vertices and edges of the planar graph.

We denote by $G_{k,k}$ the $k \times k$ grid.

Hence, for every planar graph H, there exists k_H such that $H \preccurlyeq_m G_{k_H,k_H}$, and thus $Forb_{\preccurlyeq_m}(H) \subseteq Forb_{\preccurlyeq_m}(G_{k_H,k_H})$.

Hence, the following is equivalent with the Grid Minor Theorem:

Theorem (Grid Minor Theorem)

Let k be an integer.
There exists $f(k)$ such that if $G \in Forb_{\preccurlyeq_m}(G_{k,k})$, then $tw(G) \leq f(k)$

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 as minor.

- 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 $\mathcal{P}_{i,j}$ of many disjoint $A_i - A_j$ paths avoiding the other sets.
- We then consider all the pairs $\mathcal{P}_{i,j}, \mathcal{P}_{i',j'}$
- If we can find such a pair such that many of the paths in $\mathcal{P}_{i,j}$ meets many of the path in $\mathcal{P}_{i',j'}$, 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 $\mathcal{P}_{i,j}, \mathcal{P}_{i',j'}$, many of the paths in $\mathcal{P}_{i,j}$ avoid many of the path in $\mathcal{P}_{i',j'}$.
- We can then select one path $P_{i,j} \in \mathcal{P}_{i,j}$ from each family such that these selected path are pairwise disjoint.
- Contracting each of the connected subgraph will then give us a large clique minor, which contains a large grid.

For a full proof, see section 12.4 of the book Graph Theort of Diestel.

Theorem (Grid Minor Theorem)

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

- Establishing tight bounds on $f(k)$ is an important graph-theoretical question with many applications on structural and algorithmic graph theory.

Theorem (Chekuri and Chuzhoy, 2019)

If G is $G_{k,k}$-minor free then $tw(G) < O(k^9 \, poly \log k)$.

Theorem (Grid Minor Theorem)

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

- Establishing tight bounds on $f(k)$ is an important graph-theoretical question with many applications on structural and algorithmic graph theory.
- Robertson and Seymour showed that $f(k) = \Omega(k^2 \log k)$ must hold.

Theorem (Chekuri and Chuzhoy, 2019)

If G is $G_{k,k}$-minor free then $tw(G) < O(k^9 \, poly \log k)$.

Theorem (Grid Minor Theorem)

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

- Establishing tight bounds on $f(k)$ is an important graph-theoretical question with many applications on structural and algorithmic graph theory.
- Robertson and Seymour showed that $f(k) = \Omega(k^2 \log k)$ must hold.
- For a long time, the best known upper bounds on $f(k)$ were super-exponential in k.

Theorem (Chekuri and Chuzhoy, 2019)

If G is $G_{k,k}$-minor free then $tw(G) < O(k^9 \, poly \log k)$.

Theorem (Grid Minor Theorem)

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

- Establishing tight bounds on $f(k)$ is an important graph-theoretical question with many applications on structural and algorithmic graph theory.
- Robertson and Seymour showed that $f(k) = \Omega(k^2 \log k)$ must hold.
- For a long time, the best known upper bounds on $f(k)$ were super-exponential in k.
- The first polynomial upperbound of $f(k) = O(k^{98} poly \log k)$ was proved by Chekuri and Chuzhoy in 2013.

Theorem (Chekuri and Chuzhoy, 2019)

If G is $G_{k,k}$-minor free then $tw(G) < O(k^9 poly \log k)$.

Theorem (Grid Minor Theorem)

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

- Establishing tight bounds on $f(k)$ is an important graph-theoretical question with many applications on structural and algorithmic graph theory.
- Robertson and Seymour showed that $f(k) = \Omega(k^2 \log k)$ must hold.
- For a long time, the best known upper bounds on $f(k)$ were super-exponential in k.
- The first polynomial upperbound of $f(k) = O(k^{98} poly \log k)$ was proved by Chekuri and Chuzhoy in 2013.
- Since then, many ameliorations have been proved, the best one is:

Theorem (Chekuri and Chuzhoy, 2019)

If G is $G_{k,k}$-minor free then $tw(G) < O(k^9 poly \log k)$.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN
- So we may assume that, for every $i \geq 1$, G_i is G_0-minor free.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN
- So we may assume that, for every $i \geq 1$, G_i is G_0-minor free.
- If G_0 is planar, then G_0-minor free graphs have bounded treewidth.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN
- So we may assume that, for every $i \geq 1$, G_i is G_0-minor free.
- If G_0 is planar, then G_0-minor free graphs have bounded treewidth.
- In this case $(G_1, G_2, G_3, \dots)$ satisfies Wagner Conjecture, WIN.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN
- So we may assume that, for every $i \geq 1$, G_i is G_0-minor free.
- If G_0 is planar, then G_0-minor free graphs have bounded treewidth.
- In this case $(G_1, G_2, G_3, \dots)$ satisfies Wagner Conjecture, WIN.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN
- So we may assume that, for every $i \geq 1$, G_i is G_0-minor free.
- If G_0 is planar, then G_0-minor free graphs have bounded treewidth.
- In this case $(G_1, G_2, G_3, \dots)$ satisfies Wagner Conjecture, WIN.

Corollary

The class of planar graphs satisfies Wagner Conjecture.

Planar Graphs satisfy Wagner Conjecture

Tentative proof of Wagner's Conjecture : Let $(G_n)_{n\in\mathbb{N}}$ be 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 $i \geq 1$ such that $G_0 \preccurlyeq_m G_i$, WIN
- So we may assume that, for every $i \geq 1$, G_i is G_0-minor free.
- If G_0 is planar, then G_0-minor free graphs have bounded treewidth.
- In this case $(G_1, G_2, G_3, \dots)$ satisfies Wagner Conjecture, WIN.

Corollary

The class of planar graphs satisfies Wagner Conjecture.

Would be nice to be able to say stuff on H-minor free graphs even when H is non-planar.

More precisely, Robertson and Seymour find a way to describe graphs that are H-minor-free graphs for any fixed graphs H. To do it, they use what can be called the decomposition paradigm.

2 - The Decomposition Paradigm

Introduction

The decomposition Paradigm have lead to many difficult and important results.

It is used to describe a fixed class of graphs, say $\mathcal{C}$ (in graph minor theory, classes of the form $Forb_{\preccurlyeq_m}(H)$).

The key is to describe how every graph of $\mathcal{C}$ can be constructed by gluing together certain basic graphs by a well defined composition rules.

The main result of the graph minor project is a (approximate) decomposition theorem for $Forb_{\preccurlyeq_m}(K_k)$.

This section can be seen as an introduction to the decomposition paradigm, we will show, among other things, a decomposition theorem for chordal graphs as well as for graphs of bounded treewidth.

The next section will be dedicated to perfect graphs, an illustration of what is perhaps the most dramatic success of the decomposition method (we will skip this section, no time).

Clique cutsets and treewidth

Let G be a graph and S a set of vertices of G. $G[S]$ denotes the induced subgraphs of G induced by S.
S is a **cutset** or **separator** of G if $G \setminus S$ has at least two connected components.
It is a **clique cutset** if S induces a clique (i.e. $G[S]$ is a clique).

Clique cutsets and treewidth

Let G be a graph and S a set of vertices of G. $G[S]$ denotes the induced subgraphs of G induced by S.
S is a **cutset** or **separator** of G if $G \setminus S$ has at least two connected components.
It is a **clique cutset** if S induces a clique (i.e. $G[S]$ is a clique).

The following say that we should be happy when a graph has a clique cutset.

Proposition

Let G be a graph with a clique cutset S and let $(X_i)_{i \in I}$ be the connected components of $G \setminus S$. Let $G_i = G[C_i \cup S]$. Then $\mathrm{tw}(G) = max_{i \in I}(\mathrm{tw}(G_i))$.

Clique cutsets and treewidth

Let G be a graph and S a set of vertices of G. $G[S]$ denotes the induced subgraphs of G induced by S.
S is a **cutset** or **separator** of G if $G \setminus S$ has at least two connected components.
It is a **clique cutset** if S induces a clique (i.e. $G[S]$ is a clique).

The following say that we should be happy when a graph has a clique cutset.

Proposition

Let G be a graph with a clique cutset S and let $(X_i)_{i \in I}$ be the connected components of $G \setminus S$. Let $G_i = G[C_i \cup S]$. Then $\mathrm{tw}(G) = max_{i \in I}(\mathrm{tw}(G_i))$.

Proof sketch: Take a tree decomposition (T_i, W_i) of G_i. S induces a clique, so it is contained in a bag B_i of each (T_i, W_i). Take the disjoint union of the (T_i, W_i), and add a new bag containing S and adjacent to each B_i. Prove that this is a tree decomposition of G of width $\mathrm{tw}(G) = max_{i \in I}(\mathrm{tw}(G_i))$.

Exercises on clique cutsets

Recall that $\omega(G)$ is the size of a largest clique of G.

We denote by $\omega_m(G)$ the largest k such that G contains K_k as a minor.

Exercice 1

Let G be a graph with a clique cutset S and let $(X_i)_{i \in I}$ be the connected components of $G \setminus S$. Let $G_i = G[C_i \cup S]$. Prove that:

1. $\chi(G) = max_{i \in I}(\chi(G_i))$.
2. $\omega_m(G) = max_{i \in I}(\omega_m(G_i))$

The graphs $G[C_i \cup S]$ are called the block of decomposition of G. The proposition in the previous slide and this exercise show the importance of block of decomposition.

Decomposition theorem for chordal graphs I

A graph G is **chordal** if it has no **induced subgraph** isomorphic to a cycle of length at least 4.

Chordal graphs is one of the oldest studied class of graphs. They have a very strong structure that permits to design efficient algorithms to compute on them.

Decomposition theorem for chordal graphs I

A graph G is **chordal** if it has no **induced subgraph** isomorphic to a cycle of length at least 4.

Chordal graphs is one of the oldest studied class of graphs. They have a very strong structure that permits to design efficient algorithms to compute on them.

Decomposition theorem for chordal graphs [Dirac, 1961]: If G is a chordal graph, then:

- either G is a complete graph, or
- G has a clique cutset.

Proof of Dirac theorem

Decomposition theorem for chordal graphs [Dirac, 1961]: If G is a chordal graph, then:

- either G is a complete graph, or
- G has a clique cutset.

Proof:

- Suppose that G is not a complete graph.
- Let x and y be two non-adjacent vertices. Then $V(G) \setminus \{x, y\}$ is a cutset of G separating u and v. This to say that G has some cutsets.
- Let S be a minimal vertex-cutset of G, and let C_1 and C_2 be two connected components of $G \setminus S$.
- The fact that S is minimal implies that every vertex of S has a neighbour in both C_1 and C_2.
- Suppose that $G[S]$ is not a clique.
- So S contains two non-adjacent vertices u and v.
- Since S is minimal, both u and v have a neighbor in both C_1 and C_2.
- Hence, for $i = 1, 2$, there exists an induced uv-path P_i whose interior vertices are in C_i.
- Then $P_1 \cup P_2$ induces a cycle of length at least 4, a contradiction.
- So S is a clique-cutset of G.

Decomposition theorem for chordal graph II

It is now easy to deduce the following decomposition theorem for chordal graphs.

Decomposition theorem for chordal graphs
A graph is chordal if and only if it can be constructed recursively by pasting along complete subgraphs, starting from complete graphs.

Exercice 2

Let G be a chordal graph. Prove that $\chi(G) = \omega(G)$.

Exercice 3

Let G be a chordal graph. Prove that G has a *simplicial vertex*, that is a vertex x such that $N(x) \cup \{x\}$ is a complete graph.

Decomposition theorem for chordal graph II

It is now easy to deduce the following decomposition theorem for chordal graphs.

Decomposition theorem for chordal graphs
A graph is chordal if and only if it can be constructed recursively by pasting along complete subgraphs, starting from complete graphs.

Exercice 2

Let G be a chordal graph. Prove that $\chi(G) = \omega(G)$.

Exercice 3

Let G be a chordal graph. Prove that G has a *simplicial vertex*, that is a vertex x such that $N(x) \cup \{x\}$ is a complete graph.

Hint: Among all clique cutsets S, choose one that minimize the the size of the smallest connected component C of $G \setminus S$. Prove that $G[S \cup C]$ is a complete graph, and thus all vertices in C are simplicial.

Treewidth and Chordal graphs

Given a family $\mathcal{T} = \{T_1, \ldots, T_n\}$ of trees, the intersection graph of $\mathcal{T}$ is the graph with vertices $\{v_1, \ldots, v_n\}$ such that v_i is adjacent to v_j if $V(T_i) \cap V(T_k) \neq \emptyset$.

Exercice 4

Show that the following statement are equivalent:

1. G is chordal
2. G admits a tree decomposition such that every bag is a clique.
3. G admits a tree decomposition with the property that $uv \in E(G)$ **if and only if** T_u and T_v have non-empty intersection[a] (and equivalently if and only if a bag contains both u and v).
4. G is the intersection graph of a family of subtrees of a tree[b].

Finally, use the second characterization to prove that for every graph H:

$$\mathrm{tw}(H) = \min\{\omega(G) - 1 \mid \ H \text{ subgraph of } G \text{ and } G \text{ is chordal}\}$$

Above, you may assume that G is obtained from H by adding some edges.

[a] recall that T_u is the subgraph of T induced by the node x such that $u \in W_x$ where W_x is the bag associated with x.

[b] So chordal graphs can be seen as generalisation of interval graphs

Hints for the exercise

$1 \Rightarrow 2$: Proceed by induction and use the decomposition theorem for chordal graphs.

$2 \Rightarrow 3$: If every bag is a clique, then a bag does not contain two non-adjacent vertices.

$3 \Rightarrow 4$: Let G be a graph and let $(T, W))$ a tree decomposition of G satisfying 3. For every $v \in V(G)$, let T_v the subtree of T induced by the nodes x of T such that the bag associated to x contains v. Let H be the intersection graph of $\{T_u \mid u \in V(G)\}$. We claim that $G = H$. They have the same set of vertices and, $uv \in E(G)$ if and obly if T_u and T_v intersect.

$4 \Rightarrow 1$ the intersection graphs of a family of trees cannot contain induced cycle of length at least 4 (do it when all trees are paths, it is kind of the same).

Fot the last question, see Corollary 12.3.12 of the book Graph Theory of Diestel.

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.

Similarly as for clique cutset, we have the following:

Proposition

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

And another characterization of treewidth, that is also a decomposition theorem for classes of graphs with bounded treewidth.

Theorem

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.

3 - "Proof" of Wagner Conjecture (Warning: contains major handwaving)

Proof of Wagner's Conjecture: general strategy

- Starts as before: Assume $(G_n)_{n\in\mathbb{N}}$ is a counterexample.
- We can assume that no graph G_i with $i \geq 1$ has G_0 as a minor.
- Hence $G_i \in Forb_{\preccurlyeq_m}(G_0)$ for $i \geq 1$
- Can we describe the structure of these graphs??
- It is sufficient to get a structure theorem for $Forb_{\preccurlyeq_m}(K_k)$.
- For $k \leq 4$ we have seen characterizations (small treewidth).
- For $k = 5$ there is one due to Wagner:

Wagner decomposition Theorem

Theorem (Wagner - 1937)

K_5-minor free graphs are constructed by a sequence of 3-clique sums operations starting from W_8 and planar graphs.

Wagner decomposition Theorem

Theorem (Wagner - 1937)

K_5-minor free graphs are constructed by a sequence of 3-clique sums operations starting from W_8 and planar graphs.

How to use that to prove Wagner Conjecture?
Like that:

- Assume $(G_i)_{i\in\mathbb{N}}$ is bad sequence (for every $i < j$, G_i is not a minor of G_j).
- Assume there exists $n \in \mathbb{N}$ such that such that $|V(G_n)| \leq 5$.
- Then $(G_i)_{i\geq n}$ are K_5-minor-free.
- Then we can use Wagner Theorem: the graphs G_i, $i > n$ have some kind of a 2-layer structure:
  - Outside we have a tree-like structure, which can be handled with similar methods used to handles trees (and graphs with bounded treewidth).
  - Inside (that is in the "bag" of the tree decomposition given by Wagner Theorem), graphs are planar or W_8, and we already now they are WQO.

Wagner decomposition Theorem

Theorem (Wagner - 1937)

K_5-minor free graphs are constructed by a sequence of 3-clique sums operations starting from W_8 and planar graphs.

How to use that to prove Wagner Conjecture?
Like that:

- Assume $(G_i)_{i\in\mathbb{N}}$ is bad sequence (for every $i < j$, G_i is not a minor of G_j).
- Assume there exists $n \in \mathbb{N}$ such that such that $|V(G_n)| \leq 5$.
- Then $(G_i)_{i\geq n}$ are K_5-minor-free.
- Then we can use Wagner Theorem: the graphs G_i, $i > n$ have some kind of a 2-layer structure:
  - Outside we have a tree-like structure, which can be handled with similar methods used to handles trees (and graphs with bounded treewidth).
  - Inside (that is in the "bag" of the tree decomposition given by Wagner Theorem), graphs are planar or W_8, and we already now they are WQO.

Hence, all we need is a generalisation of Wagner decomposition Theorem for all complete graphs.

Vortices and Fringes

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_i that is linked to some vertices of A_i.
- We can also add edges $v_i v_j$ if $A_i \cap A_j \neq \emptyset$.
- We call this **adding a fringe** of width k to C.

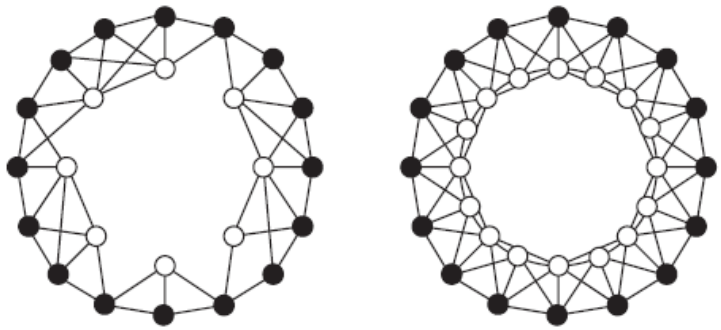


FIGURE 4. A fringe of width 2 and a fringe of width 3.

Figure of Laszló Lovász

Almost k-embeddable

Now let us define a class $\mathcal{G}_k$ of **almost k-embeddable** graphs

i Start with a surface of genus at most k and 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)

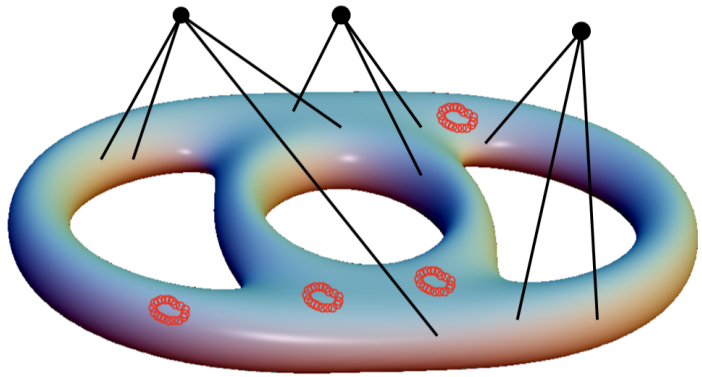
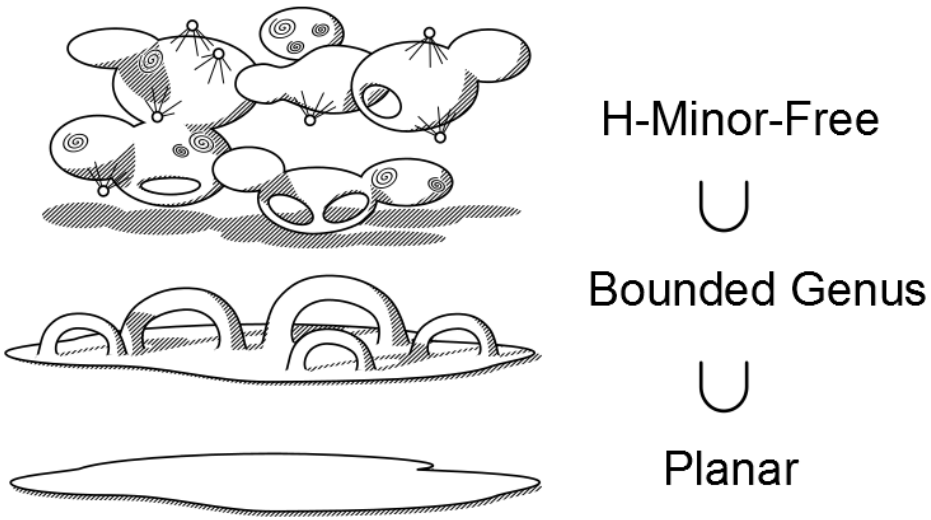


Figure by Daniel Marx

Structure Theorem

Theorem (Robertson and Seymour Theorem, XX)

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



Felix Reidl

"Proof" of Wagner Conjecture

Very (very) roughly, the proof of Wagner Conjecture is:

- Show that graphs of bounded genus satisy Wagner Conjecture 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.

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.

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?

General message:

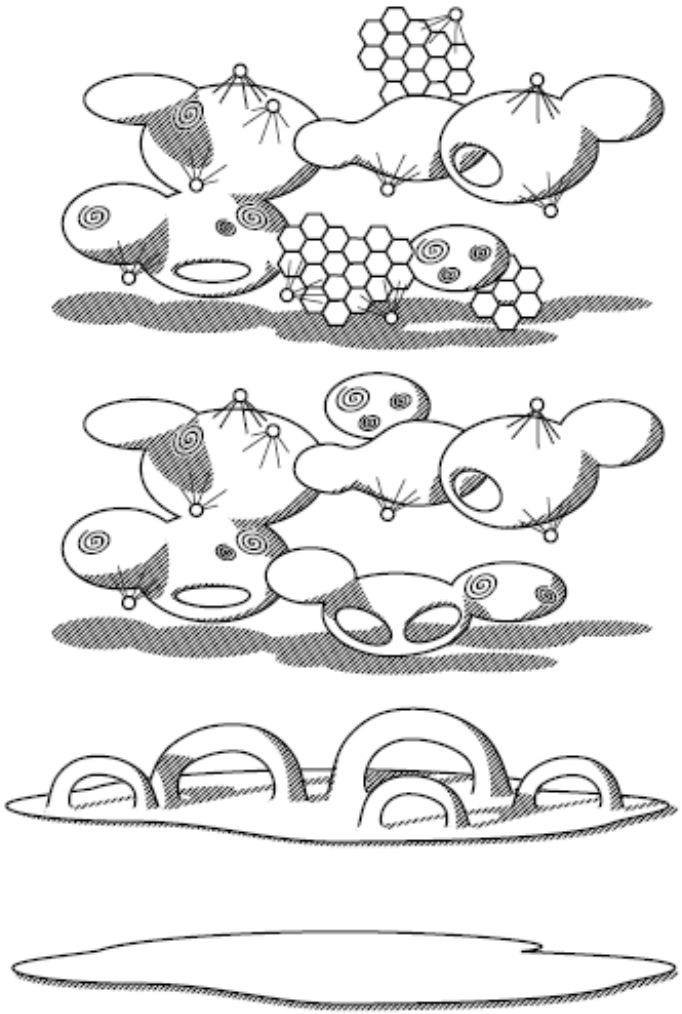
- if something works for planar graphs,
- then we might generalize it to bounded genus graphs,
- then we might generalize it to H-minor-free graphs.

What next?

What about $Forb_{\preccurlyeq_t}(H)$?

H-topological minor free graphs

H-topological minor free graphs look like that (Grohe and Marx, 2012)



H-Topological- Minor-Free

$\cup$

H-Minor-Free

$\cup$

Bounded Genus

$\cup$

Planar

Felix Reidl

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.

4 - FPT algorithm via the Graph Minor Theorem

Graph modification problem

Problem (Graph modification problem for $\mathcal{C}$)

Given: (G, k)
Question: Is there a set S of at most k vertices such that $G \setminus S \in \mathcal{C}$?

- Vertex Cover: $\mathcal{C}$ is the class of edgeless graphs.
- Feedback vertex set: $\mathcal{C}$ is the set of forest (forbidden (induce) subgraphs is the set of all cycles).
- You can take any class of graphs for $\mathcal{C}$: planar graphs, bipartite graphs, chordal graphs etc etc.

Graph modification problem

Problem (Graph modification problem for $\mathcal{C}$)

Given: (G, k)
Question: Is there a set S of at most k vertices such that $G \setminus S \in \mathcal{C}$?

- Vertex Cover: $\mathcal{C}$ is the class of edgeless graphs.
- Feedback vertex set: $\mathcal{C}$ is the set of forest (forbidden (induce) subgraphs is the set of all cycles).
- You can take any class of graphs for $\mathcal{C}$: planar graphs, bipartite graphs, chordal graphs etc etc.

We have seen that, using the branching method:

Theorem
If $\mathcal{C}$ is closed under taking induced subgraph and can be characterized by a finite set $\mathcal{F}$ of forbidden induced subgraphs (i.e. $\mathcal{C} = Forb_{\subseteq_i}(\mathcal{F})$, then the graph modification problem for $\mathcal{C}$ is FPT.

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Proof: Assume $\mathcal{C}$ is closed under taking minor, then:

- The set of YES-instance for a fixed k is also closed under taking minor (why?).

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Proof: Assume $\mathcal{C}$ is closed under taking minor, then:

- The set of YES-instance for a fixed k is also closed under taking minor (why?).
- So, for each k, there exists a set of graphs $\mathcal{F}_k$ such that the question is: does G belongs to $Forb_{\preccurlyeq_m}(\mathcal{F}_k)$?

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Proof: Assume $\mathcal{C}$ is closed under taking minor, then:

- The set of YES-instance for a fixed k is also closed under taking minor (why?).
- So, for each k, there exists a set of graphs $\mathcal{F}_k$ such that the question is: does G belongs to $Forb_{\preccurlyeq_m}(\mathcal{F}_k)$?
- By The Graph Minor Theorem, $\mathcal{F}_k$ is finite.

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Proof: Assume $\mathcal{C}$ is closed under taking minor, then:

- The set of YES-instance for a fixed k is also closed under taking minor (why?).
- So, for each k, there exists a set of graphs $\mathcal{F}_k$ such that the question is: does G belongs to $Forb_{\preccurlyeq_m}(\mathcal{F}_k)$?
- By The Graph Minor Theorem, $\mathcal{F}_k$ is finite.
- There is a $f(|F|)O(n^3)$ algorithm to decide if a graph contains a given graph F as a minor.

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Proof: Assume $\mathcal{C}$ is closed under taking minor, then:

- The set of YES-instance for a fixed k is also closed under taking minor (why?).
- So, for each k, there exists a set of graphs $\mathcal{F}_k$ such that the question is: does G belongs to $Forb_{\preccurlyeq_m}(\mathcal{F}_k)$?
- By The Graph Minor Theorem, $\mathcal{F}_k$ is finite.
- There is a $f(|F|)O(n^3)$ algorithm to decide if a graph contains a given graph F as a minor.

Graph modification problem

The following is one of the main algorithmic consequence of the graph minor theorem:

Theorem
If $\mathcal{C}$ is closed under taking minor, then the graph modification problem for $\mathcal{C}$ is FPT.

Proof: Assume $\mathcal{C}$ is closed under taking minor, then:

- The set of YES-instance for a fixed k is also closed under taking minor (why?).
- So, for each k, there exists a set of graphs $\mathcal{F}_k$ such that the question is: does G belongs to $Forb_{\preccurlyeq_m}(\mathcal{F}_k)$?
- By The Graph Minor Theorem, $\mathcal{F}_k$ is finite.
- There is a $f(|F|)O(n^3)$ algorithm to decide if a graph contains a given graph F as a minor.

Caveats:

- This is just an existential proof, we do not know how to get the set $\mathcal{F}$ of forbidden minor (the Graph Minor Theorem is not constructive)
- It is not uniform: not the same algorithm for different values of k.

5 - FPT Algorithms parametrized by treewidth: DP and Courcelle's Theorem

You've already done that with Valia. Anyway, I'll leave the slides here in case you want another presentation (though it will certainly be less good :)).

Max Weighted Independent Set

Problem (Maximum Weighted Independent Set - MWIS)

Input : A graph G with weight function $\omega : V(G) \to \mathbb{R}$
Output : an Independent set of maximum weight.

- NP-complete for general graphs.
- Polynomial for trees by Dynamic programming

MWIS for Trees with dynamic Programming

Fix a root r arbitrarily.

Denote by $ch(v)$ the set of children of v, by $T(v)$ the subtree rooted at v (hence $T(r) = T$) and set:

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

The value of a maximum weight independent set of T is precisely $f(r)$.

MWIS for Trees with dynamic Programming

Let v be a vertex of T, and let $ch(v)$ be the set of children of v. We have:

$$f^+(v) = \Sigma_{x \in ch(v)} f^-(x) + \omega(v)$$
$$f^-(v) = \Sigma_{x \in ch(v)} f(x)$$
$$f(v) = \max(f^+(v), f^-(v))$$

It only remains to compute these three functions in a bottom-up fashion (that is starting from the leaves and computing layer after layer until we rich the root), which take $O(|V(T)|)$ time.

MWIS for Trees with dynamic Programming

Let v be a vertex of T, and let $ch(v)$ be the set of children of v. We have:

$$f^+(v) = \Sigma_{x \in ch(v)} f^-(x) + \omega(v)$$
$$f^-(v) = \Sigma_{x \in ch(v)} f(x)$$
$$f(v) = \max(f^+(v), f^-(v))$$

It only remains to compute these three functions in a bottom-up fashion (that is starting from the leaves and computing layer after layer until we rich the root), which take $O(|V(T)|)$ time.

Many NP-hard problems are solvable in polytime on trees, using dynamic programming. We are going to see that the same strategy stands when applied on tree decomposition.

MWIS parametrized by treewidth

Theorem

Given a tree decomposition of width k, Maximum Weighted Independant Set can be computed in time $O(2^k \cdot k^{O(1)} \cdot n)$.

MWIS parametrized by treewidth

Theorem

Given a tree decomposition of width k, Maximum Weighted Independant Set can be computed in time $O(2^k \cdot k^{O(1)} \cdot n)$.

For each vertex $t \in V(T)$, set:
$W_t \subseteq V(G)$: vertices appearing in node t
$V_t \subseteq V(G)$: vertices appearing in the subtree rooted at t.

Generalizing the strategy used for tree:
Instead of computing two values $f^+(t)$ and $f^-(t)$, we compute $2^{|W_t|} \leq 2^k$ values for each bag W_t.

MWIS parametrized by treewidth

Theorem

Given a tree decomposition of width k, Maximum Weighted Independant Set can be computed in time $O(2^k \cdot k^{O(1)} \cdot n)$.

For each vertex $t \in V(T)$, set:
$W_t \subseteq V(G)$: vertices appearing in node t
$V_t \subseteq V(G)$: vertices appearing in the subtree rooted at t.

Generalizing the strategy used for tree:
Instead of computing two values $f^+(t)$ and $f^-(t)$, we compute $2^{|W_t|} \leq 2^k$ values for each bag W_t.

For each node t and each subset S of W_t:
$M[t, S]$ = max weighted independant I such that $I \subseteq V_t$ and $I \cap W_t = S$.

It is easy to compute $M[t, S]$ if the values are known for the children of t. But we are going to define a tree decomposition with a particular structure to ease it even more.

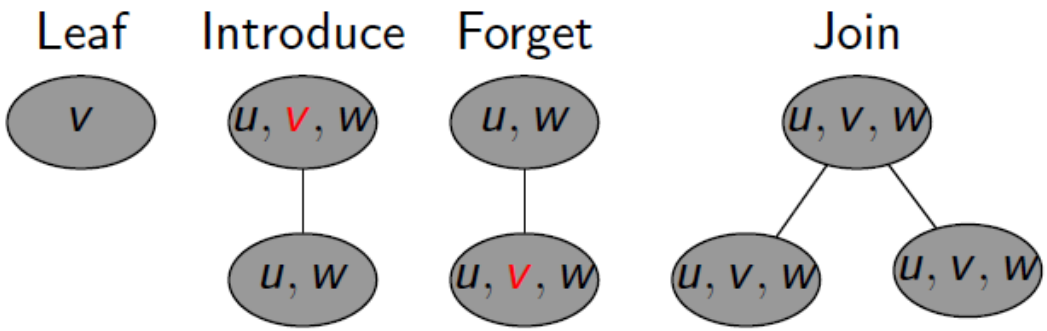
Nice Tree Decompositions

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 $|W_t| = 1$.
- Introduce: t has one child t' and $W_t = W_{t'} \setminus \{v\}$ for some $v \notin W_{t'}$
- Forget: t has one child t' and $W_t = W_t \cup \{v\}$ for some $v \notin W_t$.
- Join: t has two children t_1 and t_2 and $W_t = W_{t_1} = W_{t_2}$.



From tree decomposition to nice tree decomposition

Theorem

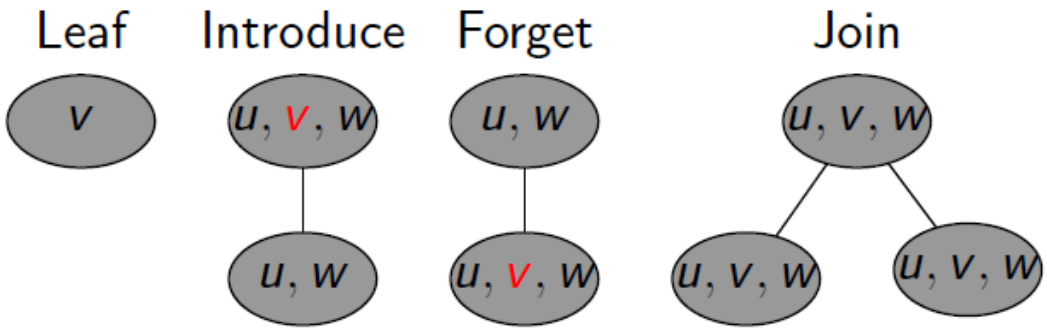
A tree decomposition of width k and n nodes can be turned into a nice tree decomposition of width k and $O(k \cdot n)$ nodes in time $O(k^2 \cdot n)$.

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.

For each node $t \in V(T)$ and each independant subset S of W_t:
$M[t, S] =$ max weighted independant I such that $I \subseteq V_t$ and $I \cap W_t = S$.



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

$$\begin{aligned} M[t, S] &= M[t', S] && \text{if } v \notin S \\ &= M[t', S \setminus \{v\}] + \omega(v) && \text{if } v \in S \end{aligned}$$

- **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 $W_t = W_{t_1} = W_{t_2}$:

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

Other Problems that are FPT by treewidth

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

Theorem

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

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

The next slides are about Courcelle's Theorem, one of the most important applications of treewidth in algorithmic theory. A super powerful meta-algorithm!

Monadic Second Order Logic

A celebrated algorithmic meta-theorem of Courcelle generalises all the previous results to monadic second order formulas.

Logical formulas on graphs are constructed inductively using

- atomic formulas : $x = y$, $v \in X$, $e \in F$ for subsets of vertices or edges.
- the binary relation $Inc(x, e)$ 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 $\exists$
- First Order formulas (FO): quantifiers over vertices and edges ($\forall v \in V(G)$; $\exists e \in E(G)$)
- MSO_1 = FO + quantify over sets of vertices,
- MSO_2 = MSO_1 + quantify over sets of edges.

Formula for 3-colorability

This is a second order formula for 3 colourability :

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

Courcelle Theorem

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

Theorem (Courcelle, 1990)

Let G be a graph and ϕ a formula of MSO_2. Assume that we are given a tree decomposition of G of width at most k. Then there is an algorithm that verify if φ is satisfied in G in time $f(|\varphi|, k) \cdot n$ for some computable function f.

Courcelle Theorem

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

Theorem (Courcelle, 1990)

Let G be a graph and ϕ a formula of MSO_2. Assume that we are given a tree decomposition of G of width at most k. Then there is an algorithm that verify if φ is satisfied in G in time $f(|\varphi|, k) \cdot n$ for some computable function f.

Note: The dependance on k can be very large (double, triple exponential etc.), therefore a direct dynamic programming algorithm can be more efficient.

If we can express a property in MSO_2, then we immediately get that testing this property is FPT parameterized by the treewidth of the input graph, as soon as we have a tree decomposition of graph at hand.

But can we compute a tree decomposition???

6 - Computing Tree Decomposition

Computing tree width

Problem

Input : A graph G and an integer k
Output : TRUE if and only if $\mathrm{tw}(G) \leq k$

Computing tree width

Problem

Input : A graph G and an integer k
Output : TRUE if and only if $\mathrm{tw}(G) \leq k$

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

Computing tree width

Problem

Input : A graph G and an integer k
Output : TRUE if and only if $\mathrm{tw}(G) \leq k$

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

Anyway, we want to use tree decomposition to design *FPT* algorithm with parameter $\mathrm{tw}(G)$, so we would be happy with time $O\left(f(\mathrm{tw}(G)) \cdot n^{0(1)}\right)$.

Computing tree width

Problem

Input : A graph G and an integer k
Output : TRUE if and only if $\mathrm{tw}(G) \leq k$

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

Anyway, we want to use tree decomposition to design *FPT* algorithm with parameter $\mathrm{tw}(G)$, so we would be happy with time $O\left(f(\mathrm{tw}(G)) \cdot n^{0(1)}\right)$.

- **FPT** : $O\left(\mathrm{tw}(G)^{\mathrm{tw}(G)^3} n\right)$ algorithm (Bodlaender, 96)

Approximate the treewidth

We don't really need to compute an *optimal tree decomposition, the following is enough.*

Theorem (Robertson and Seymour)

Given a graph G and an integer k, there is an algorithm running in time $O\left(f(k).n^2\right)$ that output:

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

This is enough for our DP based FPT algorithms: simply run the above algorithm for $k=1$, $k=2$, $k=3$,...

You are quaranted to stop for some $k \leq 4tw(G)$.

So you run $4tw(G)$ times an FPT algorithm parametrized by $tw(G)$, so you are still in time $O\left(f(tw(G)) \cdot n^{O(1)}\right)$.

And you know have a tree decomposition of width at most $4tw(G)$. Any DP based algorithm applied on it will run in time $O\left(f(tw(G)n^{O(1)}\right)$.

Computing the treewidth: state of the art

Approximation	$f(k)$	$g(n)$	reference
exact	$O(1)$	$O(n^{k+2})$	Arnborg, Corneil & Proskurowski (1987)
$4k+3$	$O(3^{3k})$	$O(n^2)$	Robertson & Seymour (1995)
$8k+7$	$2^{O(k \log k)}$	$n \log^2 n$	Lagergren (1996)
$5k+4$ (or $7k+6$)	$2^{O(k \log k)}$	$n \log n$	Reed (1992)
exact	$2^{O(k^3)}$	$O(n)$	Bodlaender (1996)
$O\left(k \cdot \sqrt{\log k}\right)$	$O(1)$	$n^{O(1)}$	Feige, Hajiaghayi & Lee (2008)
$4.5k+4$	2^{3k}	n^2	Amir (2010)
$\frac{11}{3}k+4$	$2^{3.6982k}$	$n^3 \log^4 n$	Amir (2010)
exact	$O(1)$	$O(1.7347^n)$	Fomin, Todinca & Villanger (2015)
$3k+2$	$2^{O(k)}$	$O(n \log n)$	Bodlaender et al. (2016)
$5k+4$	$2^{O(k)}$	$O(n)$	Bodlaender et al. (2016)
k^2	$O(k^7)$	$O(n \log n)$	Fomin et al. (2018)
$5k+4$	$2^{8.765k}$	$O(n \log n)$	Belbasi & Fürer (2021a)
$2k+1$	$2^{O(k)}$	$O(n)$	Korhonen (2021)
$5k+4$	$2^{6.755k}$	$O(n \log n)$	Belbasi & Fürer (2021b)
exact	$2^{O(k^2)}$	n^4	Korhonen & Lokshtanov (2022)
$(1+\varepsilon)k$	$k^{O(k/\varepsilon)}$	n^4	Korhonen & Lokshtanov (2022)

picture from wikipedia

Approximate the treewidth

Theorem

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

- *either a small certificate that $\mathrm{tw}(G) \geq k$*
- *or a tree decomposition of width at most $4k + 3$.*

The rest of this section is dedicated to design this algorithm.

See Section 7.6 of the book Parametrized Algorithms.

Good separator

Let G be a graph. A set of vertices S is a separator (or vertex cutset) if S disconnects G, that is $G \setminus S$ has at least two connected components.

Let S, X be two sets of vertices, S is a balanced X-separator if:

- S disconnects G into two parts V_1 and V_2.
- For $i = 1, 2$, V_i contains at most $2|X|/3$ vertices of X.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.
  - We know that $|A|, |B|, |C| \leq 1/2|X|$.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.
  - We know that $|A|, |B|, |C| \leq 1/2|X|$.
  - If $|A \cup B| \leq \frac{2}{3}|X|$, then we win (take $V_1 = A \cup B$ and $V_2 = C$).

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.
  - We know that $|A|, |B|, |C| \leq 1/2|X|$.
  - If $|A \cup B| \leq \frac{2}{3}|X|$, then we win (take $V_1 = A \cup B$ and $V_2 = C$).
  - Same if $|A \cup C| \leq \frac{2}{3}|X|$ or $|B \cup C| \leq \frac{2}{3}|X|$.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.
  - We know that $|A|, |B|, |C| \leq 1/2|X|$.
  - If $|A \cup B| \leq \frac{2}{3}|X|$, then we win (take $V_1 = A \cup B$ and $V_2 = C$).
  - Same if $|A \cup C| \leq \frac{2}{3}|X|$ or $|B \cup C| \leq \frac{2}{3}|X|$.
  - Simple calculation show that one of $|A \cup B|, |A \cup C|, |B \cup C|$ has at most $\frac{2}{3}|X|$ vertices.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.
  - We know that $|A|, |B|, |C| \leq 1/2|X|$.
  - If $|A \cup B| \leq \frac{2}{3}|X|$, then we win (take $V_1 = A \cup B$ and $V_2 = C$).
  - Same if $|A \cup C| \leq \frac{2}{3}|X|$ or $|B \cup C| \leq \frac{2}{3}|X|$.
  - Simple calculation show that one of $|A \cup B|, |A \cup C|, |B \cup C|$ has at most $\frac{2}{3}|X|$ vertices.

Certificate that $\mathrm{tw}(G) \geq k$

lemma: If $\mathrm{tw}(G) < k$, then every $X \subseteq V(G)$ of size at least $2k+1$ admits a balanced X-separator of size at most k

Proof Ideas:

- Take a tree decomposition (T, W) of width $k-1$ where T has maximum degree 3.
- For each node $t \in T$, the bag W_t separates G into 2 or 3 connected components.
- If one of the connected components contains more than half of the vertices of X, then orient the corresponding edge out from t.
- Prove that there exists an internal node t with no outgoing edge (look at the last vertex of maximal directed path and observe that an edge incident with a leaf is oriented out from the leaf).
- Show that W_t is a balanced X-separator with respect to X:
  - Let A, B, C be the three connected components of $G \setminus W_t$.
  - We know that $|A|, |B|, |C| \leq 1/2|X|$.
  - If $|A \cup B| \leq \frac{2}{3}|X|$, then we win (take $V_1 = A \cup B$ and $V_2 = C$).
  - Same if $|A \cup C| \leq \frac{2}{3}|X|$ or $|B \cup C| \leq \frac{2}{3}|X|$.
  - Simple calculation show that one of $|A \cup B|, |A \cup C|, |B \cup C|$ has at most $\frac{2}{3}|X|$ vertices.

Certificate that $\mathrm{tw}(G) \geq k$:
If G contains a set X of at least $2k+1$ vertices that do not admit a balanced X-separator of size a most k, then $\mathrm{tw}(G) \geq k$.

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

- If G has at most $4k$ vertices then put all vertices in a single bag.

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

- If G has at most $4k$ vertices then put all vertices in a single bag.
- If X has less than $2k + 1$ vertices, then augment X arbitrarily by adding vertices until its size is at least $2k + 1$.

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

- If G has at most $4k$ vertices then put all vertices in a single bag.
- If X has less than $2k + 1$ vertices, then augment X arbitrarily by adding vertices until its size is at least $2k + 1$.
- Assume for the moment that we know how to compute a balanced X-separator of size at most k.

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

- If G has at most $4k$ vertices then put all vertices in a single bag.
- If X has less than $2k + 1$ vertices, then augment X arbitrarily by adding vertices until its size is at least $2k + 1$.
- Assume for the moment that we know how to compute a balanced X-separator of size at most k.
- If X admits no balanced X-separator of size at most k, then by what precedes it is a certificate that $tw(G) \geq k$.

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

- If G has at most $4k$ vertices then put all vertices in a single bag.
- If X has less than $2k + 1$ vertices, then augment X arbitrarily by adding vertices until its size is at least $2k + 1$.
- Assume for the moment that we know how to compute a balanced X-separator of size at most k.
- If X admits no balanced X-separator of size at most k, then by what precedes it is a certificate that $tw(G) \geq k$.
- So we may assume that X has a balanced X-separator S of size at most k.

FPT algorithm to approx the tw

We prove by induction on the number of vertices of G the following algorithm (apply it with $X = \emptyset$ to get the desired algorithm).

Problem

Input : A graph G, an integer k and a set $X \subseteq V(G)$ such that $|X| \leq 3k$
Output : A certificate that $tw(G) \geq k$ or a rooted tree decomposition T of G of width at most $4k + 1$ where $X \subseteq root(G)$

- If G has at most $4k$ vertices then put all vertices in a single bag.
- If X has less than $2k + 1$ vertices, then augment X arbitrarily by adding vertices until its size is at least $2k + 1$.
- Assume for the moment that we know how to compute a balanced X-separator of size at most k.
- If X admits no balanced X-separator of size at most k, then by what precedes it is a certificate that $tw(G) \geq k$.
- So we may assume that X has a balanced X-separator S of size at most k.
- We are going to use it to compute the tree decomposition of width at most $4k + 1$.

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.
- Observe that $|V(G_i)| < |V(G)|$.

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.
- Observe that $|V(G_i)| < |V(G)|$.
- Apply the algorithm on (G_i, k, X_i).

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.
- Observe that $|V(G_i)| < |V(G)|$.
- Apply the algorithm on (G_i, k, X_i).
- Either certificate that $tw(G_i) \geq k$ for some i and therefore $tw(G) \geq k$

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.
- Observe that $|V(G_i)| < |V(G)|$.
- Apply the algorithm on (G_i, k, X_i).
- Either certificate that $tw(G_i) \geq k$ for some i and therefore $tw(G) \geq k$
- Or get two rooted decompositions T_1, T_2 of G_1 and G_2 with $X_i \subseteq root(T_i)$

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.
- Observe that $|V(G_i)| < |V(G)|$.
- Apply the algorithm on (G_i, k, X_i).
- Either certificate that $tw(G_i) \geq k$ for some i and therefore $tw(G) \geq k$
- Or get two rooted decompositions T_1, T_2 of G_1 and G_2 with $X_i \subseteq root(T_i)$
- Add a root bag containing all vertices in $X \cup S$ (note that $|X \cup S| \leq 4k$) attached to the roots of T_1 and T_2.

- Let S be a balanced X-separator (Recall $2k+1 \leq |X| \leq 3k$ and $|S| \leq k$)
- $G \setminus S$ disconnects G into two non-empty parts V_1 and V_2 such that $X \cap V_i \leq 2|X|/3$ for $i = 1, 2$
- Define $X_i = S \cup (X \cap V_i)$. Then $|X_i| \leq k + \frac{2}{3}3k = 3k$
- Set $G_i = G[V_i \cup S]$.
- Observe that $|V(G_i)| < |V(G)|$.
- Apply the algorithm on (G_i, k, X_i).
- Either certificate that $tw(G_i) \geq k$ for some i and therefore $tw(G) \geq k$
- Or get two rooted decompositions T_1, T_2 of G_1 and G_2 with $X_i \subseteq root(T_i)$
- Add a root bag containing all vertices in $X \cup S$ (note that $|X \cup S| \leq 4k$) attached to the roots of T_1 and T_2.
- Check that it is indeed a tree decomposition of the desired width.

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,
  - X_0 is a subset of a separator of size at most k separating V_1 and V_2 where $X_i \subseteq V_i$

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,
  - X_0 is a subset of a separator of size at most k separating V_1 and V_2 where $X_i \subseteq V_i$
- Equivalently if and only if in $G \setminus X_0$, there are at most $k - |X_0|$ disjoint paths from X_1 to X_2.

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,
  - X_0 is a subset of a separator of size at most k separating V_1 and V_2 where $X_i \subseteq V_i$
- Equivalently if and only if in $G \setminus X_0$, there are at most $k - |X_0|$ disjoint paths from X_1 to X_2.
- Ford Fulkerson do that in $O\left(k^2 n\right)$ time[1].

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,
  - X_0 is a subset of a separator of size at most k separating V_1 and V_2 where $X_i \subseteq V_i$
- Equivalently if and only if in $G \setminus X_0$, there are at most $k - |X_0|$ disjoint paths from X_1 to X_2.
- Ford Fulkerson do that in $O\left(k^2 n\right)$ time[1].
- 3^{3k} ways of defining the partition X_0, X_1, X_2 so $O\left(27^k.k^2 n\right)$ for this step

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,
  - X_0 is a subset of a separator of size at most k separating V_1 and V_2 where $X_i \subseteq V_i$
- Equivalently if and only if in $G \setminus X_0$, there are at most $k - |X_0|$ disjoint paths from X_1 to X_2.
- Ford Fulkerson do that in $O\left(k^2 n\right)$ time[1].
- 3^{3k} ways of defining the partition X_0, X_1, X_2 so $O\left(27^k.k^2 n\right)$ for this step

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

Computing the balanced separator

Here is how to compute a balanced X-separator S of size at most k in time $f(k)n^{O(1)}$:

- S exists if and only if one can partition X into three subsets X_1, X_2, X_0 such that
  - X_1 and X_2 have size at most $2|X|/3$,
  - X_0 is a subset of a separator of size at most k separating V_1 and V_2 where $X_i \subseteq V_i$
- Equivalently if and only if in $G \setminus X_0$, there are at most $k - |X_0|$ disjoint paths from X_1 to X_2.
- Ford Fulkerson do that in $O(k^2 n)$ time[1].
- 3^{3k} ways of defining the partition X_0, X_1, X_2 so $O(27^k.k^2 n)$ for this step

So the **total complexity** for the algorithm is $O(27^k.k^2 n^2)$ since the tree decomposition has at most n nodes.

[1] FF runs a number of iterations; each iteration takes $O(n+m)$ time and either concludes that the currently found flow is maximum, or augments it by 1. Since we are interested only in situations when the maximum flow is of size at most $k+1$, we may terminate the computation after $k+2$ iterations. Moreover $m \leq kn$, otherwise $\mathrm{tw}(G) > k$

7 - Win/Win approach and planar graph problems

Vertex Cover via treewidth

Observation

If $vc(G) \leq k$, then $tw(G) \leq k$

Vertex Cover via treewidth

Observation

If $vc(G) \leq k$, then $tw(G) \leq k$

Indeed, if $G - S$ is edgeless and $|S| \leq k$, then we have a path decomposition where the set of bags is $\{S \cup \{x\} \mid x \in V(G)\}$.

Vertex Cover via treewidth

Observation

If $vc(G) \leq k$, then $tw(G) \leq k$

Indeed, if $G - S$ is edgeless and $|S| \leq k$, then we have a path decomposition where the set of bags is $\{S \cup \{x\} \mid x \in V(G)\}$.

FPT algorithm for Vertex Cover (parametrized by the size of the solution):

- Run our algorithm to compute tree decomposition on (G, k).

Vertex Cover via treewidth

Observation

If $vc(G) \leq k$, then $tw(G) \leq k$

Indeed, if $G - S$ is edgeless and $|S| \leq k$, then we have a path decomposition where the set of bags is $\{S \cup \{x\} \mid x \in V(G)\}$.

FPT algorithm for Vertex Cover (parametrized by the size of the solution):

- Run our algorithm to compute tree decomposition on (G, k).
- If it ouputs that $tw(G) \geq k$, then (G, k) is a NO-instance.

Vertex Cover via treewidth

Observation

If $vc(G) \leq k$, then $tw(G) \leq k$

Indeed, if $G - S$ is edgeless and $|S| \leq k$, then we have a path decomposition where the set of bags is $\{S \cup \{x\} \mid x \in V(G)\}$.

FPT algorithm for Vertex Cover (parametrized by the size of the solution):

- Run our algorithm to compute tree decomposition on (G, k).
- If it ouputs that $tw(G) \geq k$, then (G, k) is a NO-instance.
- Otherwise we have a tree decomposition of width at most $4k + 1$ at hand.

Vertex Cover via treewidth

Observation

If $vc(G) \leq k$, then $tw(G) \leq k$

Indeed, if $G - S$ is edgeless and $|S| \leq k$, then we have a path decomposition where the set of bags is $\{S \cup \{x\} \mid x \in V(G)\}$.

FPT algorithm for Vertex Cover (parametrized by the size of the solution):

- Run our algorithm to compute tree decomposition on (G, k).
- If it ouputs that $tw(G) \geq k$, then (G, k) is a NO-instance.
- Otherwise we have a tree decomposition of width at most $4k + 1$ at hand.
- Use Dynamic Programming to compute a minimum vertex cover.

Subexponential FPT algorithm for planar graphs

See Section 7.6 of the book Parametrized Algorithms.

Grid Minor for planar graphs

We denote by $\boxplus_t$ the $t \times t$ grid.

Planar grid minor Theorem

- Every planar graph G with $tw(G) \geq 9t/2$ contains $\boxplus_t$ as a minor.

Grid Minor for planar graphs

We denote by $\boxplus_t$ the $t \times t$ grid.

Planar grid minor Theorem

- Every planar graph G with $tw(G) \geq 9t/2$ contains $\boxplus_t$ as a minor.
- Moreover, there is a $O(n^2)$-time algorithm that, given a planar graph, either output a tree-decomposition of width $9t/2$, or constructs a $\boxplus_t$-model.

Grid Minor for planar graphs

We denote by $\boxplus_t$ the $t \times t$ grid.

Planar grid minor Theorem

- Every planar graph G with $tw(G) \geq 9t/2$ contains $\boxplus_t$ as a minor.
- Moreover, there is a $O(n^2)$-time algorithm that, given a planar graph, either output a tree-decomposition of width $9t/2$, or constructs a $\boxplus_t$-model.

Corollary

Let G a planar graph on n vertices. Then:

- $tw(G) \leq \frac{9}{2}\sqrt{n+1}$ *and*
- *a tree decomposition of width $\frac{9}{2}\sqrt{n+1}$ can be constructed in $O(n^2)$ time.*

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.
- So, by the Planar Grid Minor Theorem, if $vc(G) \leq k$, then $tw(G) \leq \frac{9}{2}\sqrt{2k+2}$.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.
- So, by the Planar Grid Minor Theorem, if $vc(G) \leq k$, then $tw(G) \leq \frac{9}{2}\sqrt{2k+2}$.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.
- So, by the Planar Grid Minor Theorem, if $vc(G) \leq k$, then $tw(G) \leq \frac{9}{2}\sqrt{2k+2}$.

We now have the following algorithm:

- In $O(n^2)$, we get either a $\boxplus_{\sqrt{2k+2}}$-model, and in this cas we output NO.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.
- So, by the Planar Grid Minor Theorem, if $vc(G) \leq k$, then $tw(G) \leq \frac{9}{2}\sqrt{2k+2}$.

We now have the following algorithm:

- In $O(n^2)$, we get either a $\boxplus_{\sqrt{2k+2}}$-model, and in this cas we output NO.
- Or we get a tree decomposition of width at most $\frac{9}{2}\sqrt{2k+2}$.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.
- So, by the Planar Grid Minor Theorem, if $vc(G) \leq k$, then $tw(G) \leq \frac{9}{2}\sqrt{2k+2}$.

We now have the following algorithm:

- In $O(n^2)$, we get either a $\boxplus_{\sqrt{2k+2}}$-model, and in this cas we output NO.
- Or we get a tree decomposition of width at most $\frac{9}{2}\sqrt{2k+2}$.
- Then we use dynamic programming to computer the minimum vertex cover in time $2^{\sqrt{2k+2}} \cdot k^{0(1)} \cdot n$.

We want to solve k-Vertex Cover for an instance (G, k) where G is planar.

- Observe that $vc(\boxplus_t) = \lceil \frac{t^2}{2} \rceil$ (because it has a matching of size $\lceil \frac{t^2}{2} \rceil$).
- So if G contains $\boxplus_t$ as a minor for some $t \geq \sqrt{2k+2}$, it has no vertex cover of size k.
- So, by the Planar Grid Minor Theorem, if $vc(G) \leq k$, then $tw(G) \leq \frac{9}{2}\sqrt{2k+2}$.

We now have the following algorithm:

- In $O(n^2)$, we get either a $\boxplus_{\sqrt{2k+2}}$-model, and in this cas we output NO.
- Or we get a tree decomposition of width at most $\frac{9}{2}\sqrt{2k+2}$.
- Then we use dynamic programming to computer the minimum vertex cover in time $2^{\sqrt{2k+2}} \cdot k^{0(1)} \cdot n$.
- In total, we get an algorithm in $2^{0(\sqrt{k})} \cdot n \cdot O(n^2)$.

Subexponential parameterized algorithm

Any problem satisfying the following properties has a subexponential time FPT algorithm:

- The size of a solution in $\boxplus_k$ is of order $\Omega(k^2)$.
- Given a tree decomposition of width $O(k)$, the problem can be solved in time $O(2^k) \cdot n^{0(1)}$.
- If G has a solution of size at most k, then every minor of G too.

Dominating set

A vertex set S of a graph G is a dominating set if $S \cup N(S) = V(G)$.
In other words, every vertex has a neighbour in S or is in S.

Problem (DOMINATING SET parametrized by the size of the solution)

Question: Given (G, k), does G have a dominating set of size at most k?

Exercice 5

Can you use the graph minor theorem to prove that Dominating set parametrized by the size of the solution of FPT?

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

- A dominating set of $\boxplus_k$ has size at least $\frac{k^2}{4}$.

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

- A dominating set of $\boxplus_k$ has size at least $\frac{k^2}{4}$.
- Given a tree decomposition of width $O(k)$, we can compute a minimum dominating set in time $O(2^{O(k)}) \cdot n^{0(1)}$.

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

- A dominating set of $\boxplus_k$ has size at least $\frac{k^2}{4}$.
- Given a tree decomposition of width $O(k)$, we can compute a minimum dominating set in time $O(2^{O(k)}) \cdot n^{0(1)}$.
- But it might be that G has a smaller dominating set then one of its minor.

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

- A dominating set of $\boxplus_k$ has size at least $\frac{k^2}{4}$.
- Given a tree decomposition of width $O(k)$, we can compute a minimum dominating set in time $O(2^{O(k)}) \cdot n^{0(1)}$.
- But it might be that G has a smaller dominating set then one of its minor.

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

- A dominating set of $\boxplus_k$ has size at least $\frac{k^2}{4}$.
- Given a tree decomposition of width $O(k)$, we can compute a minimum dominating set in time $O(2^{O(k)}) \cdot n^{0(1)}$.
- But it might be that G has a smaller dominating set then one of its minor.

Indeed, deleting a vertex or even an edge, might increase a lot the size of a smallest dominating set.

Question: Does the subexponantial strategy used for vertex cover in planar graph works?

- A dominating set of $\boxplus_k$ has size at least $\frac{k^2}{4}$.
- Given a tree decomposition of width $O(k)$, we can compute a minimum dominating set in time $O(2^{O(k)}) \cdot n^{0(1)}$.
- But it might be that G has a smaller dominating set then one of its minor.

Indeed, deleting a vertex or even an edge, might increase a lot the size of a smallest dominating set.

Solution: Observe that contracting an edge can only decrease the size of a smallest dominating set, and modify the grod minor theorem!

Planar grid minor theorem for edge contraction

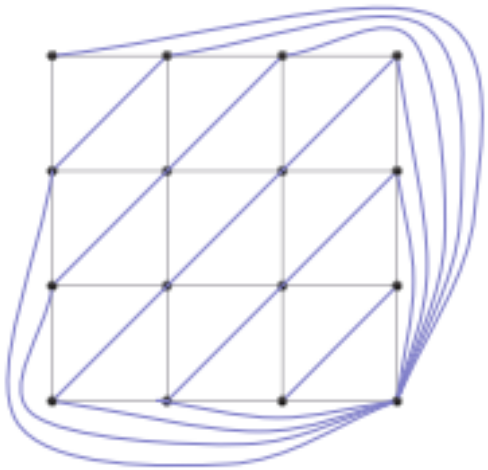
Given two graphs G and H, we say that G contains H as a contraction, if H can be obtained from G by contracting some edges.

Planar grid minor theorem for edge contraction

Given two graphs G and H, we say that G contains H as a contraction, if H can be obtained from G by contracting some edges.

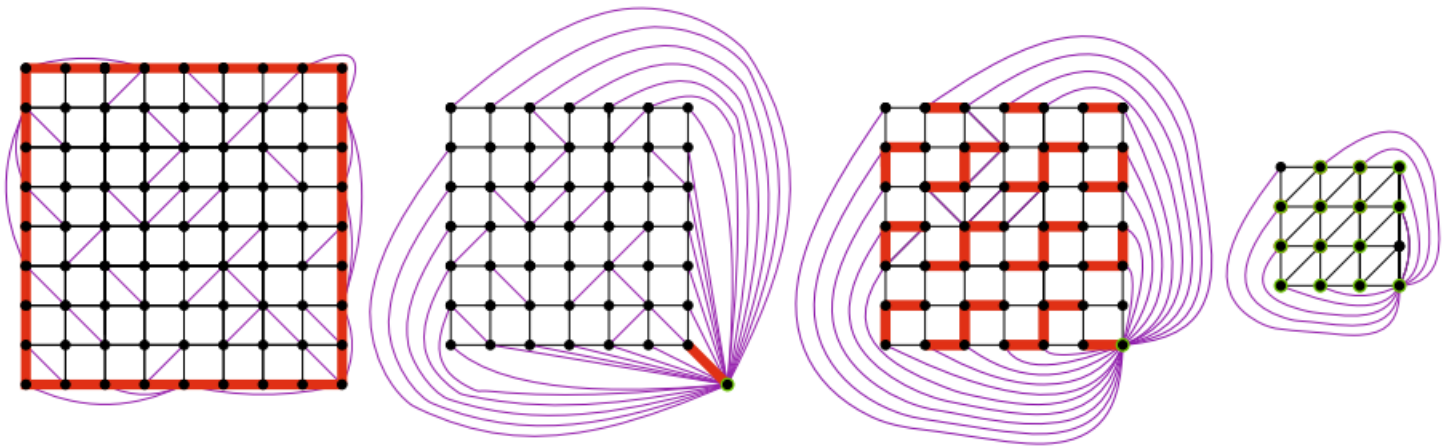
Planar grid minor theorem for edge contraction

Let G be a planar graph. If $tw(G) \geq 9t + 5$, then G contain Γ_t as a contraction. Moreover, there is an algorithm running in time $O(n^2)$ that either output a tree decomposition of width $9t + 5$, or output a set of edges whose contraction resuts in Γ_t.



Proof:

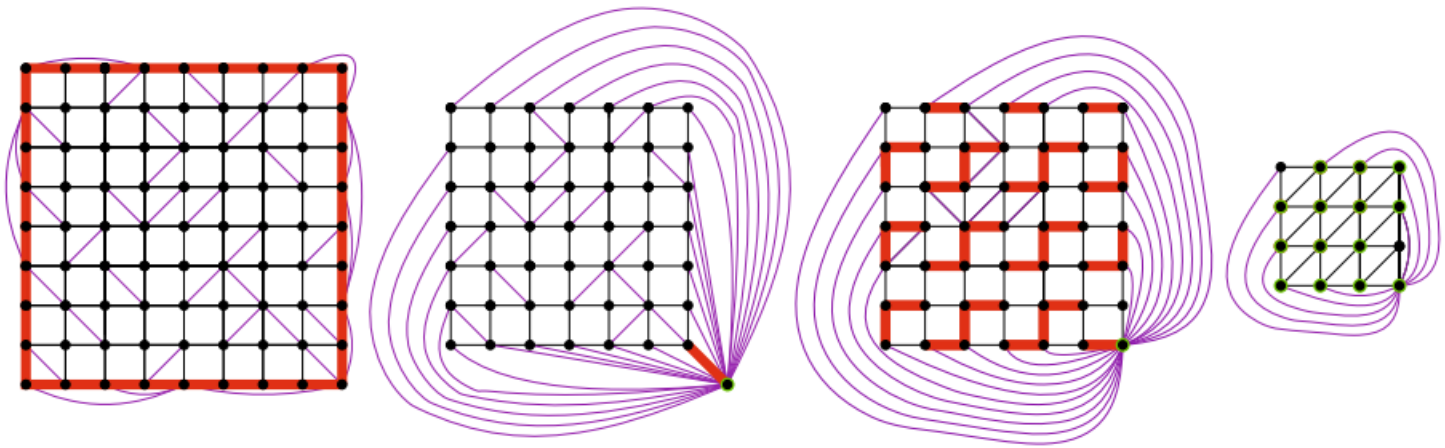
- If $tw(G) \geq 9t + 5$, then G contain a $\boxplus_{2t+1}$ model.



- Finally, the obtained Γ_t has no extra edge, since adding an edge to Γ_t spoils its planarity.

Proof:

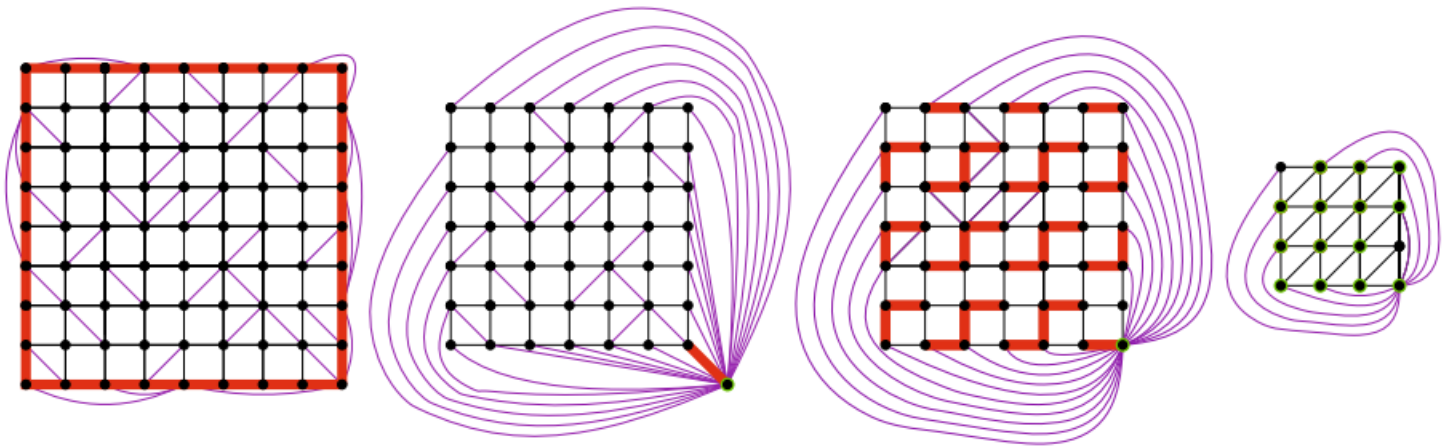
- If $tw(G) \geq 9t + 5$, then G contain a $\boxplus_{2t+1}$ model.
- Hence, after a sequence of vertex deletion, edge deletion and edge contraction, we get $\boxplus_{2t+1}$.



- Finally, the obtained Γ_t has no extra edge, since adding an edge to Γ_t spoils its planarity.

Proof:

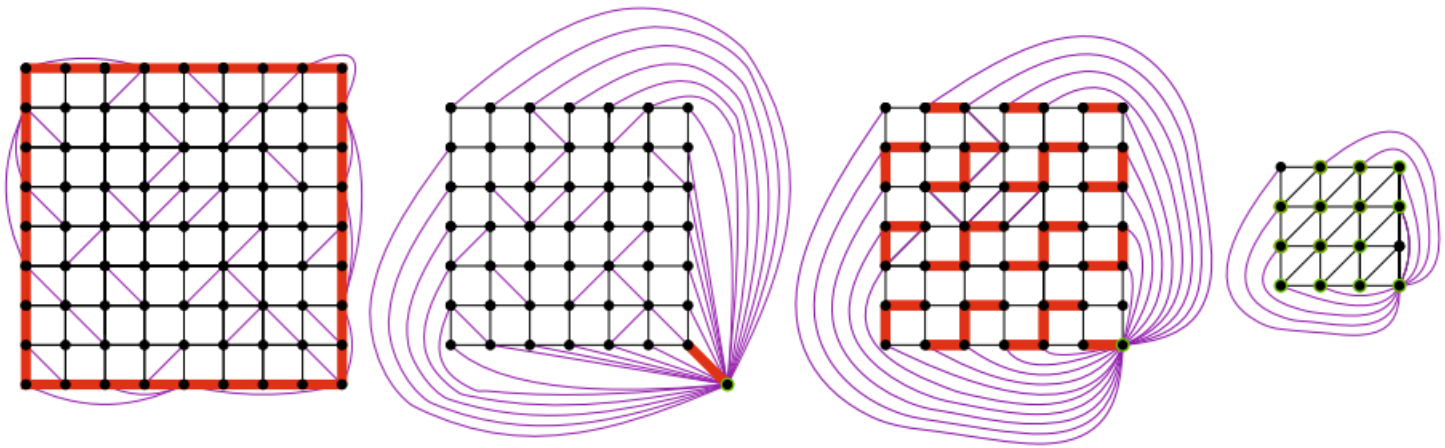
- If $tw(G) \geq 9t + 5$, then G contain a $\boxplus_{2t+1}$ model.
- Hence, after a sequence of vertex deletion, edge deletion and edge contraction, we get $\boxplus_{2t+1}$.
- Instead of deleting the vertices, contract them with one of their neighbor and omit edge deletion.



- Finally, the obtained Γ_t has no extra edge, since adding an edge to Γ_t spoils its planarity.

Proof:

- If $tw(G) \geq 9t + 5$, then G contain a $\boxplus_{2t+1}$ model.
- Hence, after a sequence of vertex deletion, edge deletion and edge contraction, we get $\boxplus_{2t+1}$.
- Instead of deleting the vertices, contract them with one of their neighbor and omit edge deletion.
- This way we get $\boxplus_{2t+1}$ plus some edges. And we get Γ_t by doing the following contradction:



- Finally, the obtained Γ_t has no extra edge, since adding an edge to Γ_t spoils its planarity.

Subexponential parameterized algorithm for dominating set

Observation

A minimum dominating set of Γ_k has size $\Omega(k^2)$.

So the strategy works again, and we get a subexponential FPT time algorithm for dominating set in planar graphs.

Subexponential parameterized algorithm for dominating set

Observation

A minimum dominating set of Γ_k has size $\Omega(k^2)$.

So the strategy works again, and we get a subexponential FPT time algorithm for dominating set in planar graphs.

General strategy:

- The size of a solution in $\boxplus_k$ is of order $\Omega(k^2)$.
- Given a tree decomposition of width $O(k)$, the problem can be solved in time $O(2^k) \cdot n^{0(1)}$.
- Contracting edges can only decrease the size of the solution.