Algorithmic graph structure theory

Dániel Marx[1]

[1]Computer and Automation Research Institute,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

STACS 2013 Tutorial
February 27, 2013
Kiel, Germany

Classes of graphs

Classes of graphs can be described by

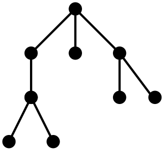
1. what they do not have,
   (excluded structures)
2. how they look like
   (constructions and decompositions).

In general, the second description is more useful for algorithmic purposes.

Classes of graphs

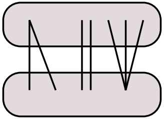
Example: Trees

1. Do not contain cycles (and connected)
2. Have a tree structure.



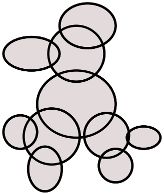
Example: Bipartite graphs

1. Do not contain odd cycles,
2. Edges going only between two classes.



Example: Chordal graphs

1. Do not contain induced cycles,
2. Clique-tree decomposition and simplicial ordering.

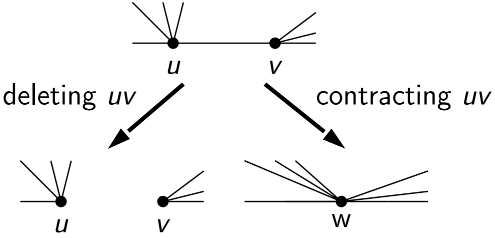


Graph Structure Theory

"Graph structure theory" usually refers to the theory developed by Robertson and Seymour on graphs excluding minors.

Definition

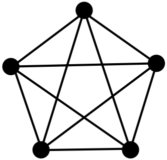
Graph H is a **minor** of G ($H \leq G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.



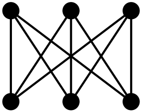
Excluding minors

Theorem [Wagner 1937]

A graph is a planar if and only if it excludes K_5 and $K_{3,3}$ as a minor.



K_5

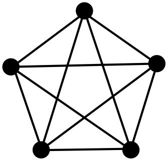


$K_{3,3}$

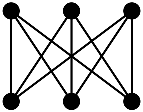
Excluding minors

Theorem [Wagner 1937]

A graph is a planar if and only if it excludes K_5 and $K_{3,3}$ as a minor.



K_5



$K_{3,3}$

- How do graphs excluding H (or $H_1, \ldots, H_k$) look like?
- What other classes can be defined this way?

The work of Robertson and Seymour gives some kind of combinatorial answer to that and provides tools for the related algorithmic questions.

Minor closed properties

Definition

A set $\mathcal{G}$ of graphs is **minor closed** if $G \in \mathcal{G}$ and $H \leq G$ implies $H \in \mathcal{G}$. .

Examples of minor closed properties:

planar graphs
graphs that can be drawn on the torus
acyclic graphs (forests)
graphs having no cycle longer than k
empty graphs

Examples of not minor closed properties:

complete graphs
regular graphs
bipartite graphs

Wagner's conjecture

Let $\mathcal{G}$ be a minor closed class of graphs. Then $\mathcal{G}$ can be characterized by the minimal **obstructions:**

Let $H \in \mathcal{F}$ if $H \notin \mathcal{G}$, but every proper minor of H is in $\mathcal{G}$.

$$G \in \mathcal{G} \iff \forall H \in \mathcal{F}, H \not\leq G$$

Wagner's conjecture

Let $\mathcal{G}$ be a minor closed class of graphs. Then $\mathcal{G}$ can be characterized by the minimal **obstructions:**

Let $H \in \mathcal{F}$ if $H \notin \mathcal{G}$, but every proper minor of H is in $\mathcal{G}$.

$$G \in \mathcal{G} \iff \forall H \in \mathcal{F}, H \not\leq G$$

Theorem [Robertson and Seymour]

Every class $\mathcal{G}$ closed under taking minors has a finite set $\mathcal{F}$ of minimal obstructions.

Graph Minors Theorem

Well-quasi-ordering:

Theorem [Robertson and Seymour]

Every class $\mathcal{G}$ closed under taking minors has a finite set $\mathcal{F}$ of minimal obstructions.

Minor testing:

Theorem [Robertson and Seymour]

For every fixed graph H, there is an $O(n^3)$ time algorithm for testing whether H is a minor of the given graph G.

Corollary: For every minor closed property $\mathcal{G}$, there is an $O(n^3)$ time algorithm for testing whether a given graph G is in $\mathcal{G}$.

Graph Minors results

- The proof spans around 400 pages in the paper series "Graph Minors I–XXIII".
- The size of the obstruction sets and the constants in the algorithms can be astronomical even for simple properties.

Graph Minors results

- The proof spans around 400 pages in the paper series "Graph Minors I–XXIII".
- The size of the obstruction sets and the constants in the algorithms can be astronomical even for simple properties.

Why should you know about this theory?

- The theory introduces simpler concepts and techniques that are useful on their own in many contexts.
- Some of the more complicated results can be formulated as self-contained powerful statements that can be used as a black box.

Graph Minors Theorem

Treewidth
Grid theorems
Planar graphs

Structure theorem

Minor testing

Well-quasi-ordering

Fixed-parameter tractability

Main definition

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Main goal of parameterized complexity: to find FPT problems.

Fixed-parameter tractability

Main definition

A parameterized problem is **fixed-parameter tractable (FPT)** if there is an $f(k)n^c$ time algorithm for some constant c.

Main goal of parameterized complexity: to find FPT problems.

Examples of NP-hard problems that are FPT:

- Finding a vertex cover of size k.
- Finding a path of length k.
- Finding k disjoint triangles.
- Drawing the graph in the plane with k edge crossings.
- Finding disjoint paths that connect k pairs of points.
- ...

Fixed-parameter tractability

- Downey and Fellows started the systematic investigation of fixed-parameter tractability and its hardness theory in the 80s.
- $n^{f(k)}$ vs. $f(k) \cdot n^c$.
- Many of the algorithmic results from graph structure theory can be formulated and appreciated using the language of fixed-parameter tractability.
- The original motivation of Downey and Fellows comes from graph structure theory!

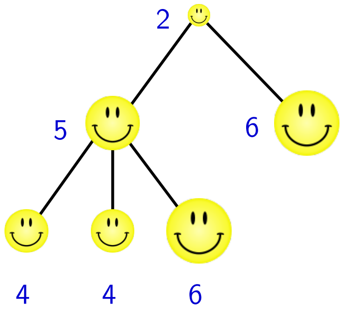
Outline

- Treewidth
  - Definition, algorithms, properties.
  - Applications
- Graphs on surfaces
- The Graph Structure Theorem
- Minor Testing
- Well-quasi-ordering

The Party Problem

Party Problem

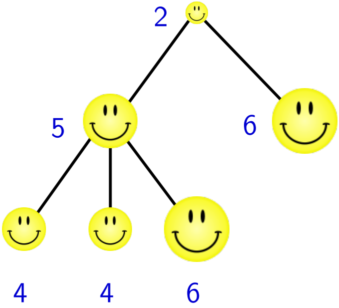
Problem:	Invite some colleagues for a party.
Maximize:	The total fun factor of the invited people.
Constraint:	Everyone should be having fun.



The Party Problem

Party Problem

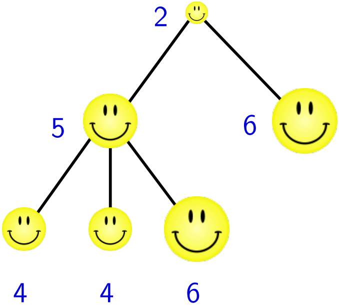
Problem:	Invite some colleagues for a party.
Maximize:	The total fun factor of the invited people.
Constraint:	Everyone should be having fun.
	Do not invite a colleague and his direct boss at the same time!



The Party Problem

PARTY PROBLEM

Problem:	Invite some colleagues for a party.
Maximize:	The total fun factor of the invited people.
Constraint:	Everyone should be having fun.
	Do not invite a colleague and his direct boss at the same time!

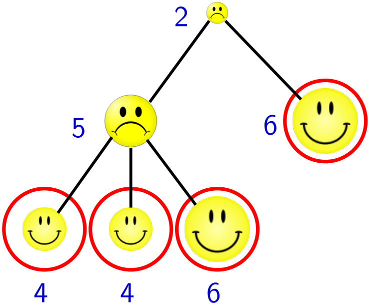


- **Input:** A tree with weights on the vertices.
- **Task:** Find an independent set of maximum weight.

The Party Problem

Party Problem

Problem: Invite some colleagues for a party.
Maximize: The total fun factor of the invited people.
Constraint: Everyone should be having fun.
Do not invite a colleague and
his direct boss at the same time!



- **Input:** A tree with weights on the vertices.
- **Task:** Find an independent set of maximum weight.

Solving the Party Problem

Dynamic programming paradigm:
We solve a large number of subproblems that depend on each other. The answer is a single subproblem.

Subproblems:

T_v: the subtree rooted at v.
$A[v]$: max. weight of an independent set in T_v
$B[v]$: max. weight of an independent set in T_v
that does not contain v

Goal: determine $A[r]$ for the root r.

Solving the Party Problem

Subproblems:

T_v: the subtree rooted at v.

$A[v]$: max. weight of an independent set in T_v

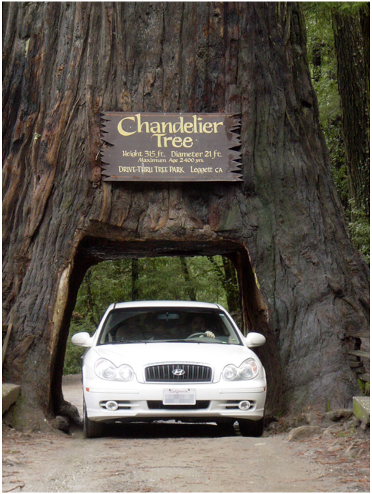
$B[v]$: max. weight of an independent set in T_v that does not contain v

Recurrence:

Assume $v_1, \ldots, v_k$ are the children of v. Use the recurrence relations

$$\boxed{\begin{array}{l} B[v] = \sum_{i=1}^{k} A[v_i] \\ A[v] = \max\{B[v]\ ,\ w(v) + \sum_{i=1}^{k} B[v_i]\} \end{array}}$$

The values $A[v]$ and $B[v]$ can be calculated in a bottom-up order (the leaves are trivial).

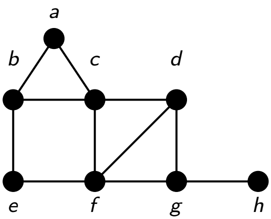

Chandelier Tree
Height 315 ft. Diameter 21 ft.
Maximum Age 2400 yrs.
DRIVE-THRU TREE PARK Leggett CA

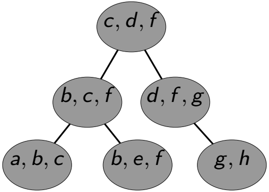
Treewidth

Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

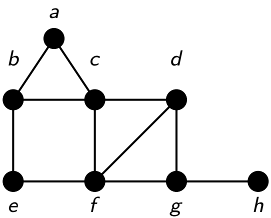


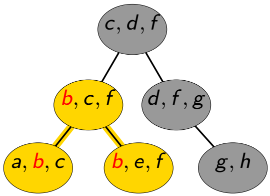


Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

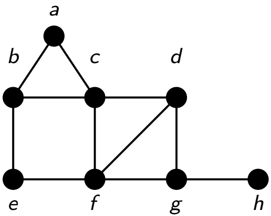


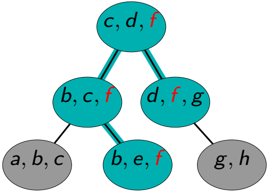


Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.





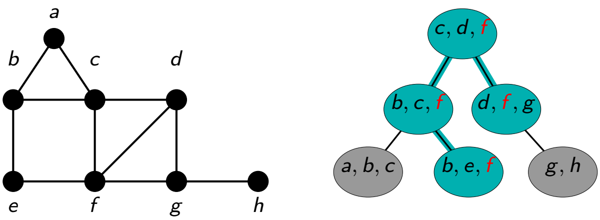
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



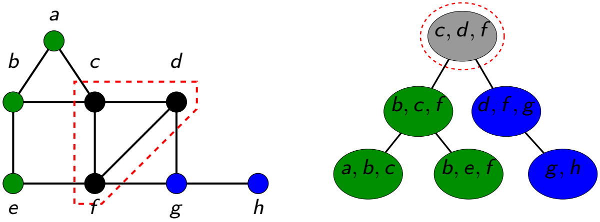
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



Each bag is a separator.

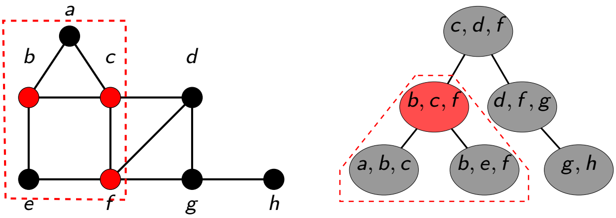
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

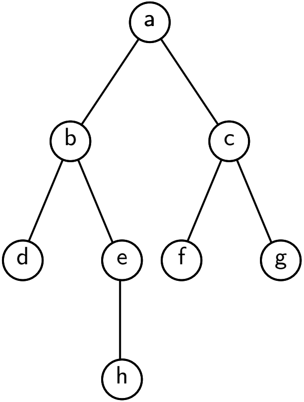
treewidth: width of the best decomposition.



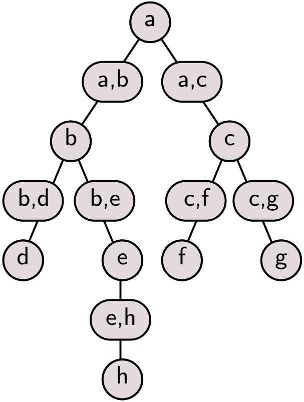
A subtree communicates with the outside world only via the root of the subtree.

Treewidth

Fact: treewidth $= 1 \iff$ graph is a forest



$\Rightarrow$



Exercise: A cycle cannot have a tree decomposition of width 1.

Treewidth — outline

1. Basic algorithms
2. Combinatorial properties
3. Applications

Finding tree decompositions

Hardness:

Theorem [Arnborg, Corneil, Proskurowski 1987]

It is NP-hard to determine the treewidth of a graph (given a graph G and an integer w, decide if the treewidth of G is at most w).

Fixed-parameter tractability:

Theorem [Bodlaender 1996]

There is a $2^{O(w^3)} \cdot n$ time algorithm that finds a tree decomposition of width w (if exists).

Consequence:
If we want an FPT algorithm parameterized by treewidth w of the input graph, then we can assume that a tree decomposition of width w is available.

Finding tree decompositions — approximately

Sometimes we can get better dependence on treewidth using approximation.

FPT approximation:

Theorem [Robertson and Seymour]

There is a $O(3^{3w} \cdot w \cdot n^2)$ time algorithm that finds a tree decomposition of width $4w + 1$, if the treewidth of the graph is at most w.

Polynomial-time approximation:

Theorem [Feige, Hajiaghayi, Lee 2008]

There is a polynomial-time algorithm that finds a tree decomposition of width $O(w\sqrt{\log w})$, if the treewidth of the graph is at most w.

Weighted Max Independent Set and treewidth

Theorem

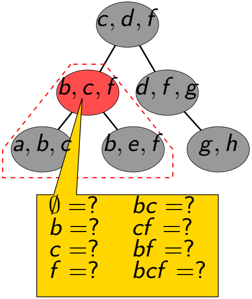
Given a tree decomposition of width w, Weighted Max Independent Set can be solved in time $O(2^w \cdot w^{O(1)} \cdot n)$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

Generalizing our solution for trees:

Instead of computing 2 values $A[v]$, $B[v]$ for each **vertex** of the graph, we compute $2^{|B_x|} \leq 2^{w+1}$ values for each bag B_x.

$M[x, S]$:
the max. weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.



Weighted Max Independent Set and treewidth

Theorem

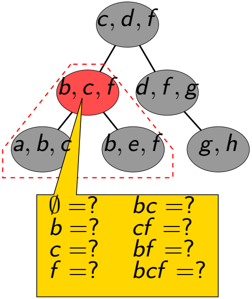
Given a tree decomposition of width w, Weighted Max Independent Set can be solved in time $O(2^w \cdot w^{O(1)} \cdot n)$.

B_x: vertices appearing in node x.

V_x: vertices appearing in the subtree rooted at x.

Generalizing our solution for trees:

Instead of computing 2 values $A[v]$, $B[v]$ for each **vertex** of the graph, we compute $2^{|B_x|} \leq 2^{w+1}$ values for each bag B_x.



$M[x, S]$:
the max. weight of an independent set $I \subseteq V_x$ with $I \cap B_x = S$.

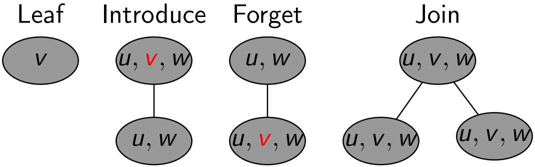
How to determine $M[x, S]$ if all the values are known for the children of x?

Nice tree decompositions

Definition

A rooted tree decomposition is **nice** if every node x is one of the following 4 types:

- **Leaf:** no children, $|B_x| = 1$
- **Introduce:** 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v
- **Forget:** 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v
- **Join:** 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$



Nice tree decompositions

Definition

A rooted tree decomposition is **nice** if every node x is one of the following 4 types:

- **Leaf:** no children, $|B_x| = 1$
- **Introduce:** 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v
- **Forget:** 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v
- **Join:** 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$

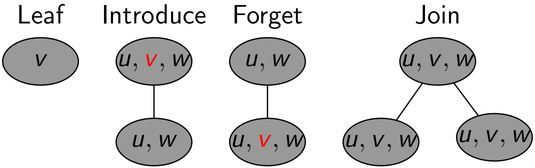
Theorem

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

Weighted Max Independent Set
and nice tree decompositions

- **Leaf:** no children, $|B_x| = 1$
  Trivial!
- **Introduce:** 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

$$m[x,S] = \begin{cases} m[y,S] & \text{if } v \notin S, \\ m[y, S \setminus \{v\}] + w(v) & \text{if } v \in S \text{ but } v \text{ has no neighbor in } S, \\ -\infty & \text{if } S \text{ contains } v \text{ and its neighbor.} \end{cases}$$



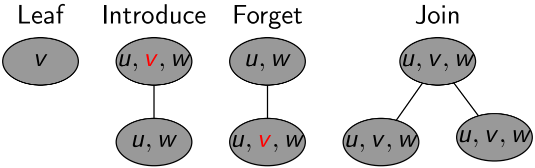
Weighted Max Independent Set
and nice tree decompositions

- **Forget:** 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v

$$m[x, S] = \max\{m[y, S], m[y, S \cup \{v\}]\}$$

- **Join:** 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$

$$m[x, S] = m[y_1, S] + m[y_2, S] - w(S)$$



Weighted Max Independent Set
and nice tree decompositions

- **Forget:** 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v

$$m[x,S] = \max\{m[y,S], m[y,S \cup \{v\}]\}$$

- **Join:** 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$

$$m[x,S] = m[y_1,S] + m[y_2,S] - w(S)$$

There are at most $2^{w+1} \cdot n$ subproblems $m[x,S]$ and each subproblem can be solved in $w^{O(1)}$ time (assuming the children are already solved).

$\Downarrow$

Running time is $O(2^w \cdot w^{O(1)} \cdot n)$.

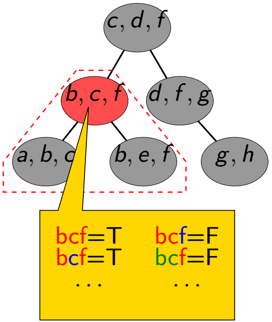
3-Coloring and tree decompositions

Theorem

Given a tree decomposition of width w, 3-Coloring can be solved in $O(3^w \cdot w^{O(1)} \cdot n)$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

For every node x and coloring $c : B_x \to \{1, 2, 3\}$, we compute the Boolean value $E[x, c]$, which is true if and only if c can be extended to a proper 3-coloring of V_x.



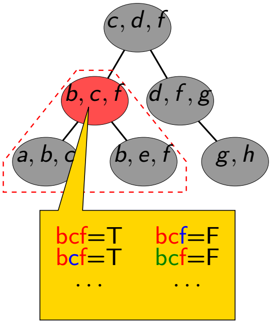
3-Coloring and tree decompositions

Theorem

Given a tree decomposition of width w, 3-Coloring can be solved in $O(3^w \cdot w^{O(1)} \cdot n)$.

B_x: vertices appearing in node x.
V_x: vertices appearing in the subtree rooted at x.

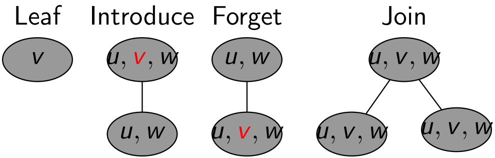
For every node x and coloring $c : B_x \to \{1, 2, 3\}$, we compute the Boolean value $E[x, c]$, which is true if and only if c can be extended to a proper 3-coloring of V_x.



How to determine $E[x, c]$ if all the values are known for the children of x?

3-Coloring and nice tree decompositions

- **Leaf:** no children, $|B_x| = 1$
  Trivial!
- **Introduce:** 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v
  If $c(v) \neq c(u)$ for every neighbor u of v, then
  $E[x, c] = E[y, c']$, where c' is c restricted to B_y.
- **Forget:** 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v
  $E[x, c]$ is true if $E[y, c']$ is true for one of the 3 extensions of c to B_y.
- **Join:** 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$
  $E[x, c] = E[y_1, c] \wedge E[y_2, c]$



3-Coloring and nice tree decompositions

- **Leaf:** no children, $|B_x| = 1$
  Trivial!
- **Introduce:** 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v
  If $c(v) \neq c(u)$ for every neighbor u of v, then
  $E[x,c] = E[y,c']$, where c' is c restricted to B_y.
- **Forget:** 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v
  $E[x,c]$ is true if $E[y,c']$ is true for one of the 3 extensions of c to B_y.
- **Join:** 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$
  $E[x,c] = E[y_1,c] \wedge E[y_2,c]$

There are at most $3^{w+1} \cdot n$ subproblems $E[x,c]$ and each subproblem can be solved in $w^{O(1)}$ time (assuming the children are already solved).

$\Rightarrow$ Running time is $O(3^w \cdot w^{O(1)} \cdot n)$.

$\Rightarrow$ 3-Coloring is FPT parameterized by treewidth.

Hamiltonian cycle and treewidth

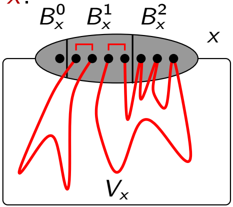
Theorem

Given a tree decomposition of width w, HAMILTONIAN CYCLE can be solved in time $w^{O(w)} \cdot n$.

B_x: vertices appearing in node x.

V_x: vertices appearing in the subtree rooted at x.

If H is a Hamiltonian cycle, then the subgraph $H[V_x]$ is a set of paths with endpoints in B_x.



What are the important properties of $H[V_x]$ "seen from outside"?

- The subsets B_x^0, B_x^1, B_x^2 of B_x having degree 0, 1, and 2.
- The matching M of B_x^1.

No. of subproblems (B_x^0, B_x^1, B_x^2, M) for node x: at most $3^w \cdot w^w$.

Hamiltonian cycle and nice tree decompositions

For each subproblem (B_x^0, B_x^1, B_x^2, M), we have to determine if there is a set of paths with this pattern.

How to do this for the different types of nodes?
(Assuming that all the subproblems are solved for the children.)

Leaf: no children, $|B_x| = 1$

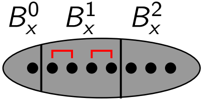
Trivial!

Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Forget: 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v

In a solution H of (B_x^0, B_x^1, B_x^2, M), vertex v has degree 2. Thus subproblem (B_x^0, B_x^1, B_x^2, M) of x is equivalent to subproblem $(B_x^0, B_x^1, B_x^2 \cup \{v\}, M)$ of y.

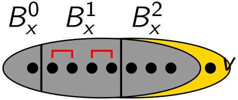


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Forget: 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v

In a solution H of (B_x^0, B_x^1, B_x^2, M), vertex v has degree 2. Thus subproblem (B_x^0, B_x^1, B_x^2, M) of x is equivalent to subproblem $(B_x^0, B_x^1, B_x^2 \cup \{v\}, M)$ of y.

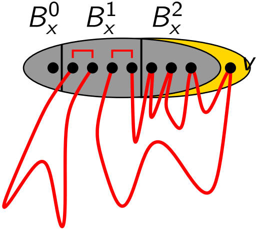


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Forget: 1 child y with $B_x = B_y \setminus \{v\}$ for some vertex v

In a solution H of (B_x^0, B_x^1, B_x^2, M), vertex v has degree 2. Thus subproblem (B_x^0, B_x^1, B_x^2, M) of x is equivalent to subproblem $(B_x^0, B_x^1, B_x^2 \cup \{v\}, M)$ of y.

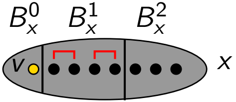


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 1: $v \in B_x^0$. Subproblem is equivalent with $(B_x^0 \setminus \{v\}, B_x^1, B_x^2, M)$ for node y.

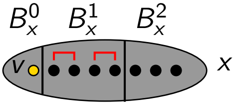


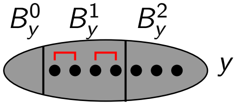
Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 1: $v \in B_x^0$. Subproblem is equivalent with $(B_x^0 \setminus \{v\}, B_x^1, B_x^2, M)$ for node y.



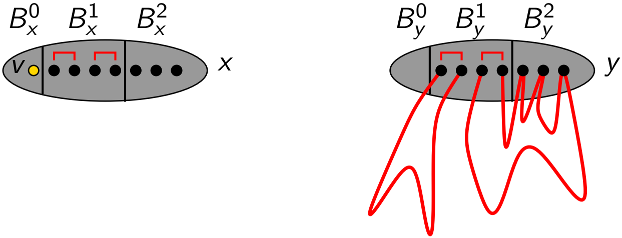


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 1: $v \in B_x^0$. Subproblem is equivalent with $(B_x^0 \setminus \{v\}, B_x^1, B_x^2, M)$ for node y.

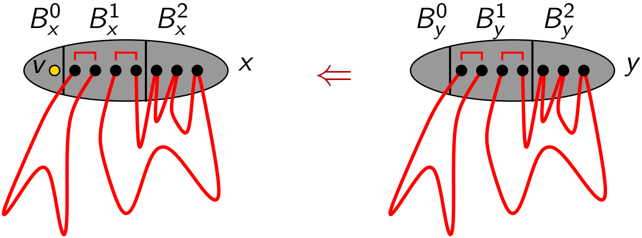


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 1: $v \in B_x^0$. Subproblem is equivalent with $(B_x^0 \setminus \{v\}, B_x^1, B_x^2, M)$ for node y.



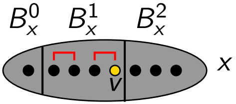
Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 2: $v \in B_x^1$. Every neighbor of v in V_x is in B_x. Thus v has to be adjacent with one other vertex of B_x.

Is there a subproblem $(B_y^0, B_y^1, B_y^2, M')$ of node y such that making a vertex of B_y adjacent to v makes it a solution for subproblem (B_x^0, B_x^1, B_x^2, M) of node x?



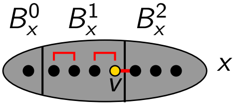
Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 2: $v \in B_x^1$. Every neighbor of v in V_x is in B_x. Thus v has to be adjacent with one other vertex of B_x.

Is there a subproblem $(B_y^0, B_y^1, B_y^2, M')$ of node y such that making a vertex of B_y adjacent to v makes it a solution for subproblem (B_x^0, B_x^1, B_x^2, M) of node x?



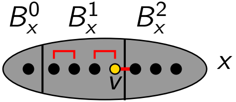
Hamiltonian cycle and nice tree decompositions

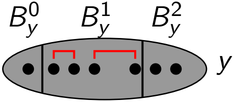
Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 2: $v \in B_x^1$. Every neighbor of v in V_x is in B_x. Thus v has to be adjacent with one other vertex of B_x.

Is there a subproblem $(B_y^0, B_y^1, B_y^2, M')$ of node y such that making a vertex of B_y adjacent to v makes it a solution for subproblem (B_x^0, B_x^1, B_x^2, M) of node x?





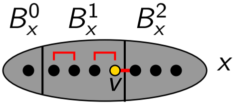
Hamiltonian cycle and nice tree decompositions

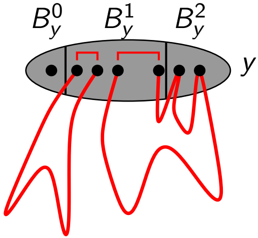
Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 2: $v \in B_x^1$. Every neighbor of v in V_x is in B_x. Thus v has to be adjacent with one other vertex of B_x.

Is there a subproblem $(B_y^0, B_y^1, B_y^2, M')$ of node y such that making a vertex of B_y adjacent to v makes it a solution for subproblem (B_x^0, B_x^1, B_x^2, M) of node x?





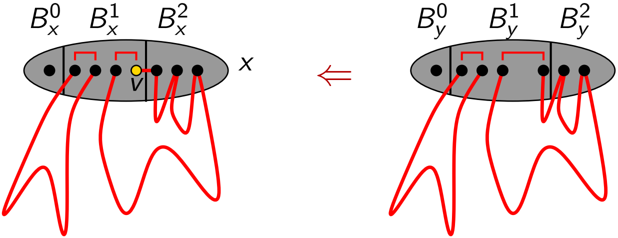
Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 2: $v \in B_x^1$. Every neighbor of v in V_x is in B_x. Thus v has to be adjacent with one other vertex of B_x.

Is there a subproblem $(B_y^0, B_y^1, B_y^2, M')$ of node y such that making a vertex of B_y adjacent to v makes it a solution for subproblem (B_x^0, B_x^1, B_x^2, M) of node x?

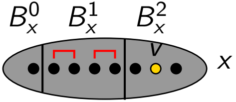


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 3: $v \in B_x^1$. Similar to Case 2, but 2 vertices of B_y are adjacent with v.

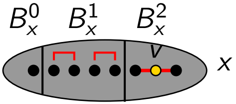


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 3: $v \in B_x^1$. Similar to Case 2, but 2 vertices of B_y are adjacent with v.



Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

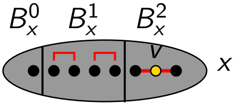
Case 3: $v \in B_x^1$. Similar to Case 2, but 2 vertices of B_y are adjacent with v.

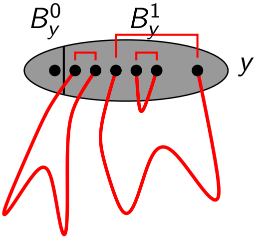
Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 3: $v \in B_x^1$. Similar to Case 2, but 2 vertices of B_y are adjacent with v.



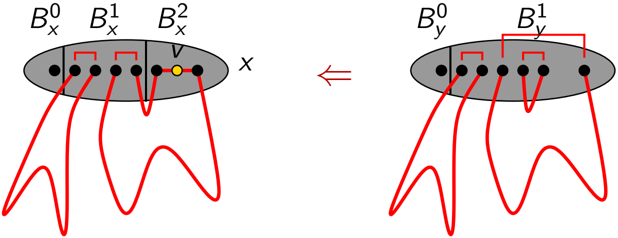


Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Introduce: 1 child y with $B_x = B_y \cup \{v\}$ for some vertex v

Case 3: $v \in B_x^1$. Similar to Case 2, but 2 vertices of B_y are adjacent with v.



Hamiltonian cycle and nice tree decompositions

Solving subproblem (B_x^0, B_x^1, B_x^2, M) of node x.

Join: 2 children y_1, y_2 with $B_x = B_{y_1} = B_{y_2}$

A solution H is the union of a subgraph $H_1 \subseteq G[V_{y_1}]$ and a subgraph $H_2 \subseteq G[V_{y_2}]$.

If H_1 is a solution for $(B_{y_1}^0, B_{y_1}^1, B_{y_1}^2, M_1)$ of node y_1 and H_2 is a solution for $(B_{y_2}^0, B_{y_2}^1, B_{y_2}^2, M_2)$ of node y_2, then we can check if $H_1 \cup H_2$ is a solution for (B_x^0, B_x^1, B_x^2, M) of node x.

For any two subproblems of y_1 and y_2, we check if they have solutions and if their union is a solution for (B_x^0, B_x^1, B_x^2, M) of node x.

Monadic Second Order Logic

Extended Monadic Second Order Logic (EMSO)

A logical language on graphs consisting of the following:

- Logical connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$, $=$, $\neq$
- quantifiers $\forall$, $\exists$ over vertex/edge variables
- predicate $\mathrm{adj}(u, v)$: vertices u and v are adjacent
- predicate $\mathrm{inc}(e, v)$: edge e is incident to vertex v
- quantifiers $\forall$, $\exists$ over vertex/edge set variables
- $\in$, $\subseteq$ for vertex/edge sets

Example:
The formula

$$\exists C \subseteq V \exists v_0 \in C \forall v \in C\ \exists u_1, u_2 \in C (u_1 \neq u_2 \wedge \mathrm{adj}(u_1, v) \wedge \mathrm{adj}(u_2, v))$$

is true on graph G if and only if ...

Monadic Second Order Logic

Extended Monadic Second Order Logic (EMSO)

A logical language on graphs consisting of the following:

- Logical connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$, $=$, $\neq$
- quantifiers $\forall$, $\exists$ over vertex/edge variables
- predicate $\mathrm{adj}(u, v)$: vertices u and v are adjacent
- predicate $\mathrm{inc}(e, v)$: edge e is incident to vertex v
- quantifiers $\forall$, $\exists$ over vertex/edge set variables
- $\in$, $\subseteq$ for vertex/edge sets

Example:
The formula

$$\exists C \subseteq V \exists v_0 \in C \forall v \in C \; \exists u_1, u_2 \in C (u_1 \neq u_2 \wedge \mathrm{adj}(u_1, v) \wedge \mathrm{adj}(u_2, v))$$

is true on graph G if and only if G has a cycle.

Courcelle's Theorem

Courcelle's Theorem

If a graph property can be expressed in EMSO, then for every fixed $w \geq 1$, there is a linear-time algorithm for testing this property on graphs having treewidth at most w.

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

Courcelle's Theorem

Courcelle's Theorem

If a graph property can be expressed in EMSO, then for every fixed $w \geq 1$, there is a linear-time algorithm for testing this property on graphs having treewidth at most w.

Note: The constant depending on w 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 EMSO, then we immediately get that testing this property is FPT parameterized by the treewidth w of the input graph.

Can we express 3-Coloring and Hamiltonian Cycle in EMSO?

Using Courcelle's Theorem

3-Coloring

$\exists C_1, C_2, C_3 \subseteq V \left(\forall v \in V \left(v \in C_1 \vee v \in C_2 \vee v \in C_3\right)\right) \wedge \left(\forall u, v \in V \operatorname{adj}(u, v) \rightarrow \left(\neg(u \in C_1 \wedge v \in C_1) \wedge \neg(u \in C_2 \wedge v \in C_2) \wedge \neg(u \in C_3 \wedge v \in C_3)\right)\right)$

Using Courcelle's Theorem

3-Coloring

$\exists C_1, C_2, C_3 \subseteq V \big(\forall v \in V\, (v \in C_1 \vee v \in C_2 \vee v \in C_3)\big) \wedge \big(\forall u, v \in V\, \mathrm{adj}(u, v) \rightarrow (\neg(u \in C_1 \wedge v \in C_1) \wedge \neg(u \in C_2 \wedge v \in C_2) \wedge \neg(u \in C_3 \wedge v \in C_3))\big)$

Hamiltonian Cycle

$\exists H \subseteq E \big(\mathrm{spanning}(H) \wedge (\forall v \in V\, \mathrm{degree2}(H, v))\big)$

$\mathrm{degree0}(H, v) := \neg\exists e \in H\, \mathrm{inc}(e, v)$

$\mathrm{degree1}(H, v) := \neg\mathrm{degree0}(H, v) \wedge \big(\neg\exists e_1, e_2 \in H\, (e_1 \neq e_2 \wedge \mathrm{inc}(e_1, v) \wedge \mathrm{inc}(e_2, v))\big)$

$\mathrm{degree2}(H, v) := \neg\mathrm{degree0}(H, v) \wedge \neg\mathrm{degree1}(H, v) \wedge \big(\neg\exists e_1, e_2, e_3 \in H\, (e_1 \neq e_2 \wedge e_2 \neq e_3 \wedge e_1 \neq e_3 \wedge \mathrm{inc}(e_1, v) \wedge \mathrm{inc}(e_2, v) \wedge \mathrm{inc}(e_3, v))\big)$

$\mathrm{spanning}(H) := \forall u, v \in V\, \exists P \subseteq H\, \forall x \in V\, \big(((x = u \vee x = v) \wedge \mathrm{degree1}(P, x)) \vee (x \neq u \wedge x \neq v \wedge (\mathrm{degree0}(P, x) \vee \mathrm{degree2}(P, x)))\big)$

Using Courcelle's Theorem

Two ways of using Courcelle's Theorem:

1. The problem can be described by a single formula (e.g, 3-Coloring, Hamiltonian Cycle).

   $\Rightarrow$ Problem can be solved in time $f(w) \cdot n$ for graphs of treewidth at most w, i.e., FPT parameterized by treewidth.

Using Courcelle's Theorem

Two ways of using Courcelle's Theorem:

1. The problem can be described by a single formula (e.g, 3-Coloring, Hamiltonian Cycle).

   $\Rightarrow$ Problem can be solved in time $f(w) \cdot n$ for graphs of treewidth at most w, i.e., FPT parameterized by treewidth.

2. The problem can be described by a formula for each value of the parameter k.

   Example: For each k, having a cycle of length exactly k can be expressed as

   $$\exists v_1, \dots, v_k \in V \, ((v_1 \neq v_2) \wedge (v_1 \neq v_3) \wedge \dots (v_{k-1} \neq v_k)) \\ \wedge (\mathrm{adj}(v_1, v_2) \wedge \mathrm{adj}(v_2, v_3) \wedge \dots \wedge \mathrm{adj}(v_{k-1}, v_k) \wedge \mathrm{adj}(v_k, v_1)).$$

   $\Rightarrow$ Problem can be solved in time $f(k, w) \cdot n$ for graphs of treewidth w, i.e., FPT parameterized with combined parameter k and treewidth w.

Subgraph Isomorphism

Subgraph Isomorphism

Input: graphs H and G
Find: a copy of H in G as subgraph.

Subgraph Isomorphism

Subgraph Isomorphism

Input: graphs H and G
Find: a copy of H in G as subgraph.

For each H, we can construct a formula ϕ_H that expresses "G has a subgraph isomorphic to H" (similarly to the k-cycle on the previous slide).

$\Rightarrow$ By Courcelle's Theorem, Subgraph Isomorphism can be solved in time $f(H, w) \cdot n$ if G has treewidth at most w.

Subgraph Isomorphism

Subgraph Isomorphism

Input: graphs H and G
Find: a copy of H in G as subgraph.

Since there is only a finite number of simple graphs on k vertices, Subgraph Isomorphism can be solved in time $f(k, w) \cdot n$ if H has k vertices and G has treewidth at most w.

Theorem

Subgraph Isomorphism is FPT parameterized by combined parameter $k := |V(H)|$ and the treewidth w of G.

MSO on words

Theorem [Büchi, Elgot, Trakhtenbrot 1960]

If a language $L \subseteq \Sigma^*$ can be defined by an MSO formula ϕ using the relation $<$, then L is regular.

Example: a^*bc^* is defined by

$$\exists x : P_b(x) \wedge (\forall y : (y < x) \to P_a(y)) \wedge (\forall y : (x < y) \to P_c(y)).$$

MSO on words

Theorem [Büchi, Elgot, Trakhtenbrot 1960]

If a language $L \subseteq \Sigma^*$ can be defined by an MSO formula ϕ using the relation $<$, then L is regular.

Example: a^*bc^* is defined by

$$\exists x : P_b(x) \wedge (\forall y : (y < x) \rightarrow P_a(y)) \wedge (\forall y : (x < y) \rightarrow P_c(y)).$$

We prove a more general statement for formulas $\phi(w, X_1, \ldots, X_k)$ and words over $\Sigma \cup \{0,1\}^k$.

Induction over the structure of ϕ:

- FSM for $\neg\phi(w)$, given FSM for $\phi(w)$.
- FSM for $\phi_1(w) \wedge \phi_2(w)$, given FSMs for $\phi_1(w)$ and $\phi_2(w)$.
- FSM for $\exists X \phi(w, X)$, given FSM for $\phi(w, X)$.
- etc.

MSO on words

Theorem [Büchi, Elgot, Trakhtenbrot 1960]

If a language $L \subseteq \Sigma^*$ can be defined by an MSO formula ϕ using the relation $<$, then L is regular.

Proving Courcelle's Theorem:

- Generalize from words to trees.
- A width-k tree decomposition can be interpreted as a tree over an alphabet of size $f(k)$.
- Formula $\Rightarrow$ tree automata.

Algorithms — overview

- Algorithms exploit the fact that a subtree communicates with the rest of the graph via a single bag.
- Key point: defining the subproblems.
- Courcelle's Theorem makes this process automatic for many problems.
- There are notable problems that are easy for trees, but hard for bounded-treewidth graphs.

Treewidth — outline

1. Basic algorithms
2. Combinatorial properties
3. Applications

Properties of treewidth

Fact: Treewidth does not increase if we delete edges, delete vertices, or contract edges.

$\Rightarrow$ If F is a **minor** of G, then the treewidth of F is at most the treewidth of G.

Properties of treewidth

Fact: Treewidth does not increase if we delete edges, delete vertices, or contract edges.

$\Rightarrow$ If F is a **minor** of G, then the treewidth of F is at most the treewidth of G.

Fact: For every clique K, there is a bag B with $K \subseteq B$.

Fact: The treewidth of the k-clique is $k-1$.

Properties of treewidth

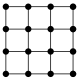
Fact: Treewidth does not increase if we delete edges, delete vertices, or contract edges.

$\Rightarrow$ If F is a **minor** of G, then the treewidth of F is at most the treewidth of G.

Fact: For every clique K, there is a bag B with $K \subseteq B$.

Fact: The treewidth of the k-clique is $k-1$.

Fact: For every $k \geq 2$, the treewidth of the $k \times k$ grid is exactly k.



The Cops and Robber game

Game: k cops try to capture a robber in the graph.

- In each step, the cops can move from vertex to vertex arbitrarily with helicopters.
- The robber moves infinitely fast on the edges, and sees where the cops will land.

Theorem [Seymour and Thomas 1993]

$k+1$ cops can win the game $\iff$ the treewidth of the graph is at most k.

The Cops and Robber game

Game: k cops try to capture a robber in the graph.

- In each step, the cops can move from vertex to vertex arbitrarily with helicopters.
- The robber moves infinitely fast on the edges, and sees where the cops will land.

Theorem [Seymour and Thomas 1993]

$k+1$ cops can win the game $\iff$ the treewidth of the graph is at most k.

Consequence 1: Algorithms

The winner of the game can be determined in time $n^{O(k)}$ using standard techniques (there are at most n^k positions for the cops)

$$\Downarrow$$

For every fixed k, it can be checked in polynomial-time if treewidth is at most k.

The Cops and Robber game

Game: k cops try to capture a robber in the graph.

- In each step, the cops can move from vertex to vertex arbitrarily with helicopters.
- The robber moves infinitely fast on the edges, and sees where the cops will land.

Theorem [Seymour and Thomas 1993]

$k+1$ cops can win the game $\iff$ the treewidth of the graph is at most k.

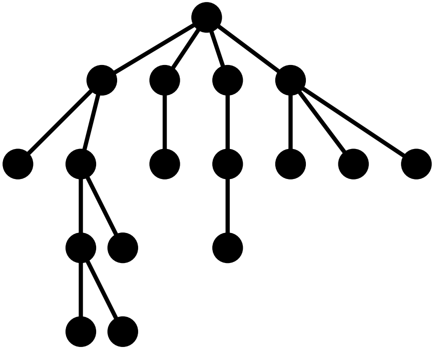
Consequence 2: Lower bounds

Exercise 1:
Show that the treewidth of the $k \times k$ grid is at least $k-1$.
(E.g., robber can win against $k-1$ cops.)

Exercise 2:
Show that the treewidth of the $k \times k$ grid is at least k.
(E.g., robber can win against k cops.)

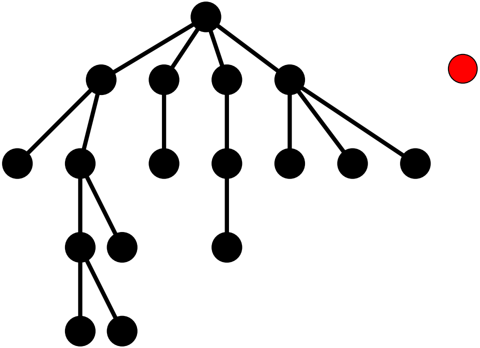
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



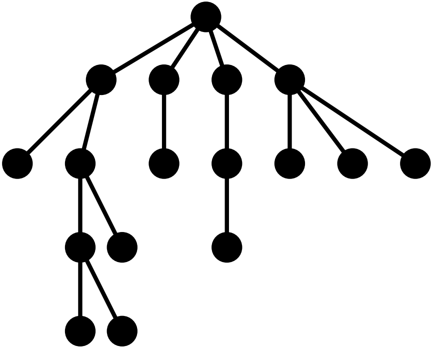
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



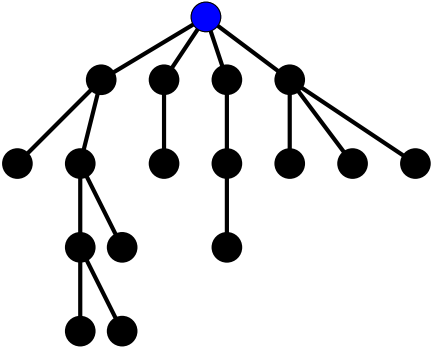
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



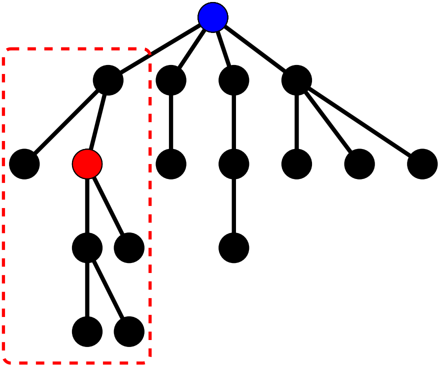
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



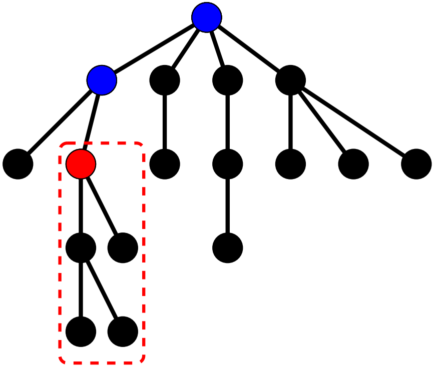
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



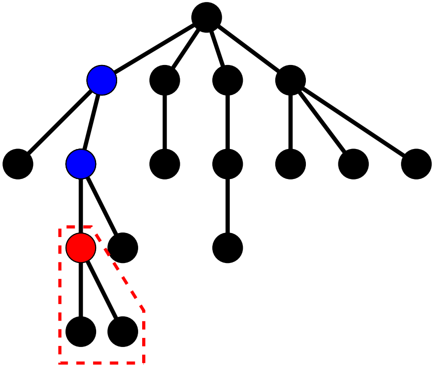
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



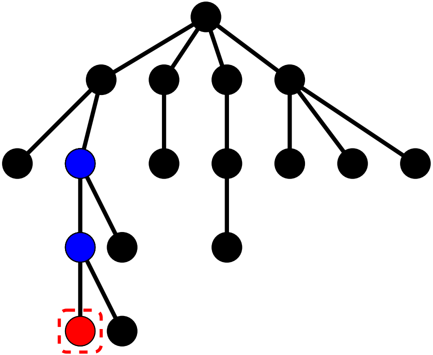
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



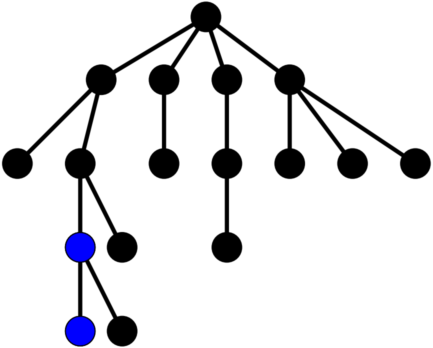
The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.



The Cops and Robber game

Example: 2 cops have a winning strategy in a tree.

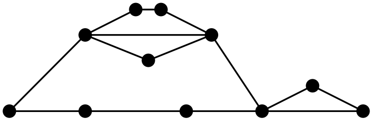


A perfect structure theorem

Theorem

The following are equivalent:

- G does not have a K_4 minor.
- G has treewidth ≤ 2.
- G is subgraph of a series-parallel graph.



A perfect structure theorem

Theorem

The following are equivalent:

- G does not have a K_4 minor.
- G has treewidth ≤ 2.
- G is subgraph of a series-parallel graph.

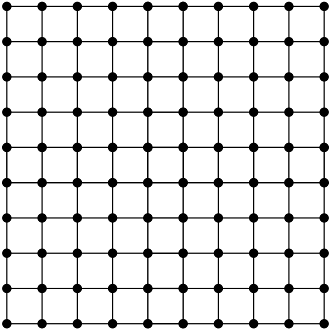
A perfect structure theorem:

$K_4 \not\in G$	$\Longrightarrow$	G has a width-2 tree decomposition
G has a width-2 tree decomposition	$\Longrightarrow$	$K_4 \not\in G$

Excluded Grid Theorem

Excluded Grid Theorem [Diestel et al. 1999]

If the treewidth of G is at least $k^{4k^2(k+2)}$, then G has a $k \times k$ grid minor.



Excluded Grid Theorem

Excluded Grid Theorem [Diestel et al. 1999]

If the treewidth of G is at least $k^{4k^2(k+2)}$, then G has a $k \times k$ grid minor.

Observation: Every planar graph is the minor of a sufficiently large grid.

Consequence

If H is planar, then every H-minor free graph has treewidth at most $f(H)$.

Excluded Grid Theorem

Excluded Grid Theorem [Diestel et al. 1999]

If the treewidth of G is at least $k^{4k^2(k+2)}$, then G has a $k \times k$ grid minor.

A large grid minor is a "witness" that treewidth is large, but the relation is approximate:

No $k \times k$ grid minor	$\Longrightarrow$	tree decomposition of width $< f(k)$
tree decomposition of width $< f(k)$	$\Longrightarrow$	no $f(k) \times f(k)$ grid minor

Excluding trees

As every forest (tree) is planar we have have for every forest F

no F-minor	$\Longrightarrow$	tree decomposition of width $< f(F)$
tree decomposition of width $< f(F)$	$\Longrightarrow$	Does not exclude any tree as minor!

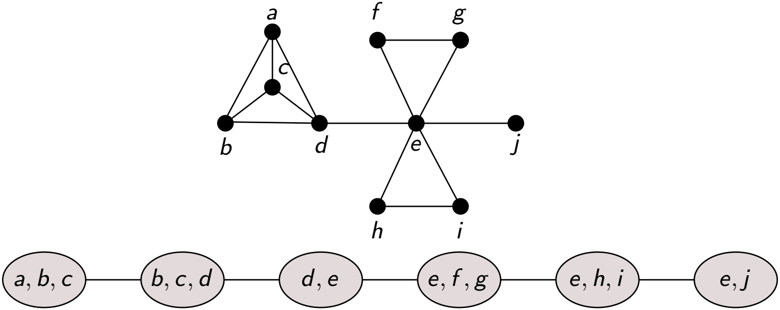
This is not a good (approximate) structure theorem.

Excluding trees

Path decomposition: the tree of bags is a path.
Pathwidth: defined analogously to treewidth.
Example: A complete binary tree on k levels has pathwidth $k-1$.

Theorem [Diestel 1995]

If F is a forest, then every F-minor free graph has pathwidth at most $|V(F)|-2$.



Excluding trees

Path decomposition: the tree of bags is a path.
Pathwidth: defined analogously to treewidth.
Example: A complete binary tree on k levels has pathwidth $k-1$.

Theorem [Diestel 1995]

If F is a forest, then every F-minor free graph has pathwidth at most $|V(F)|-2$.

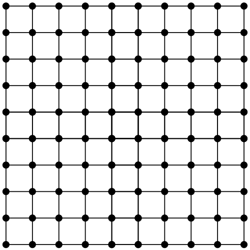
no F-minor	$\Longrightarrow$	path decomposition of width $< f(F)$
path decomposition of width $< f(F)$	$\Longrightarrow$	No $(f(F)+1)$-level complete binary tree

Planar Excluded Grid Theorem

For planar graphs, we get linear instead of exponential dependence:

Theorem [Robertson, Seymour, Thomas 1994]

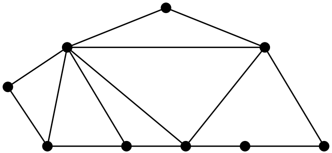
Every **planar graph** with treewidth at least $4k$ has a $k \times k$ grid minor.



Outerplanar graphs

Definition

A planar graph is **outerplanar** if it has a planar embedding where every vertex is on the infinite face.



Fact

Every outerplanar graph has treewidth at most 2.

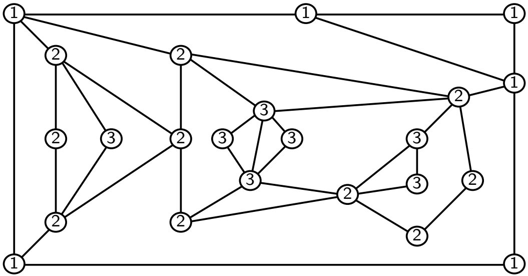
$\Rightarrow$ Every outerplanar graph is subgraph of a series-parallel graph.

k-outerplanar graphs

Given a planar embedding, we can define **layers** by iteratively removing the vertices on the infinite face.

Definition

A planar graph is k-**outerplanar** if it has a planar embedding having at most k layers.



Fact

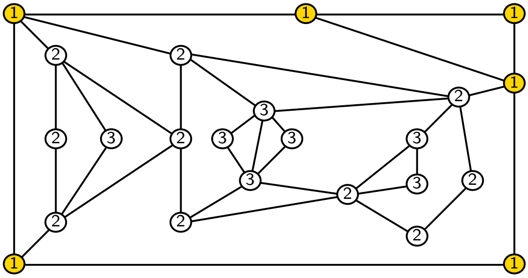
Every k-outerplanar graph has treewidth at most $3k + 1$.

k-outerplanar graphs

Given a planar embedding, we can define **layers** by iteratively removing the vertices on the infinite face.

Definition

A planar graph is k-**outerplanar** if it has a planar embedding having at most k layers.



Fact

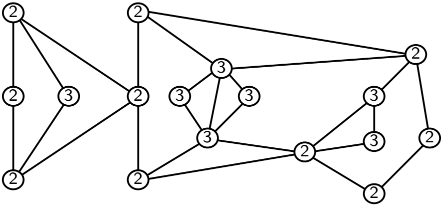
Every k-outerplanar graph has treewidth at most $3k + 1$.

k-outerplanar graphs

Given a planar embedding, we can define **layers** by iteratively removing the vertices on the infinite face.

Definition

A planar graph is k-**outerplanar** if it has a planar embedding having at most k layers.



Fact

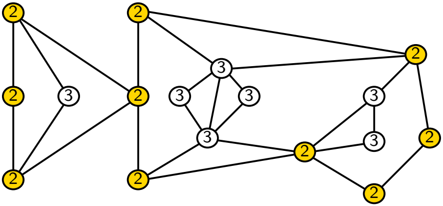
Every k-outerplanar graph has treewidth at most $3k + 1$.

k-outerplanar graphs

Given a planar embedding, we can define **layers** by iteratively removing the vertices on the infinite face.

Definition

A planar graph is k-**outerplanar** if it has a planar embedding having at most k layers.



Fact

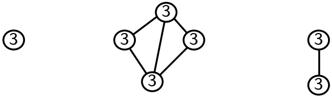
Every k-outerplanar graph has treewidth at most $3k + 1$.

k-outerplanar graphs

Given a planar embedding, we can define **layers** by iteratively removing the vertices on the infinite face.

Definition

A planar graph is k-**outerplanar** if it has a planar embedding having at most k layers.



Fact

Every k-outerplanar graph has treewidth at most $3k + 1$.

Treewidth — outline

1. Basic algorithms
2. Combinatorial properties
3. Applications
    - The shifting technique
    - Bidimensionality
    - Complexity of CSP

Approximation schemes

Definition

A **polynomial-time approximation scheme (PTAS)** for a problem P is an algorithm that takes an instance of P and a rational number $\epsilon > 0$,

- always finds a $(1+\epsilon)$-approximate solution,
- the running time is polynomial in n for every fixed $\epsilon > 0$.

Typical running times: $2^{1/\epsilon} \cdot n$, $n^{1/\epsilon}$, $(n/\epsilon)^2$, n^{1/ϵ^2}.

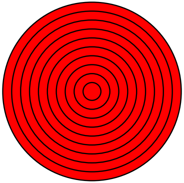
Some classical problems that have a PTAS:

- Independent Set for planar graphs
- TSP in the Euclidean plane
- Steiner Tree in planar graphs
- Knapsack

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for Independent Set for planar graphs.

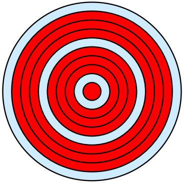


- Let $D := 1/\epsilon$. For a fixed $0 \le s < D$, delete every layer L_i with $i = s \pmod D$

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for Independent Set for planar graphs.

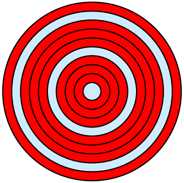


- Let $D := 1/\epsilon$. For a fixed $0 \leq s < D$, delete every layer L_i with $i = s \pmod{D}$

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for INDEPENDENT SET for planar graphs.

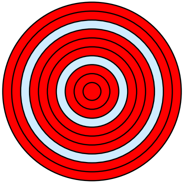


- Let $D := 1/\epsilon$. For a fixed $0 \leq s < D$, delete every layer L_i with $i = s \pmod{D}$

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for INDEPENDENT SET for planar graphs.

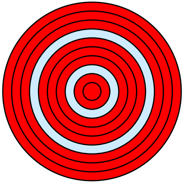


- Let $D := 1/\epsilon$. For a fixed $0 \leq s < D$, delete every layer L_i with $i = s \pmod D$

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for INDEPENDENT SET for planar graphs.

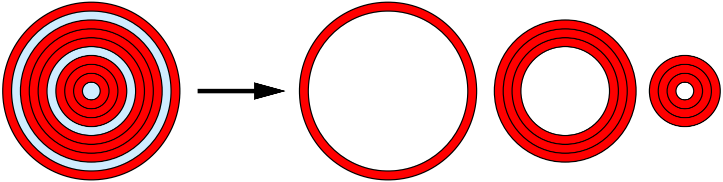


- Let $D := 1/\epsilon$. For a fixed $0 \leq s < D$, delete every layer L_i with $i = s \pmod D$

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for INDEPENDENT SET for planar graphs.

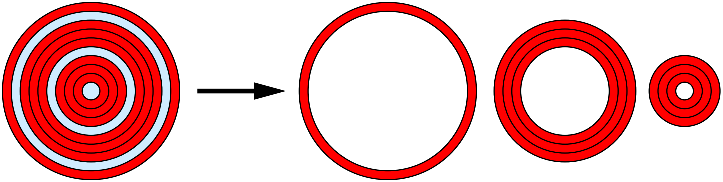


- Let $D := 1/\epsilon$. For a fixed $0 \leq s < D$, delete every layer L_i with $i = s \pmod{D}$
- The resulting graph is D-outerplanar, hence it has treewidth at most $3D + 1 = O(1/\epsilon)$.
- Using the $2^{O(\text{tw})} \cdot n$ time algorithm for INDEPENDENT SET, the problem on the D-outerplanar graph can be solved in time $2^{O(1/\epsilon)} \cdot n$.

Baker's shifting strategy for PTAS

Theorem

There is a $2^{O(1/\epsilon)} \cdot n$ time PTAS for INDEPENDENT SET for planar graphs.



We do this for every $0 \leq s < D$:
for at least one value of s, we delete
at most $1/D = \epsilon$ fraction of the solution

$\Downarrow$

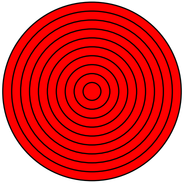
We get a $(1 + \epsilon)$-approximate solution.

Baker's shifting strategy for FPT

SUBGRAPH ISOMORPHISM

Input: graphs H and G

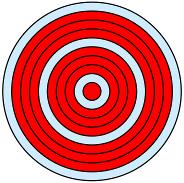
Find: a copy of H in G as subgraph.



Baker's shifting strategy for FPT

Subgraph Isomorphism

Input: graphs H and G
Find: a copy of H in G as subgraph.

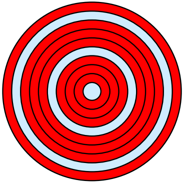


- For a fixed $0 \leq s < k + 1$, delete every layer L_i with $i = s \pmod{k+1}$

Baker's shifting strategy for FPT

SUBGRAPH ISOMORPHISM

Input: graphs H and G
Find: a copy of H in G as subgraph.

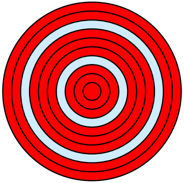


- For a fixed $0 \leq s < k+1$, delete every layer L_i with $i = s \pmod{k+1}$

Baker's shifting strategy for FPT

SUBGRAPH ISOMORPHISM

Input: graphs H and G
Find: a copy of H in G as subgraph.



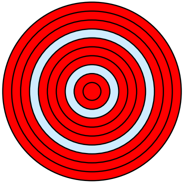
- For a fixed $0 \leq s < k+1$, delete every layer L_i with $i = s \pmod{k+1}$

Baker's shifting strategy for FPT

SUBGRAPH ISOMORPHISM

Input: graphs H and G

Find: a copy of H in G as subgraph.



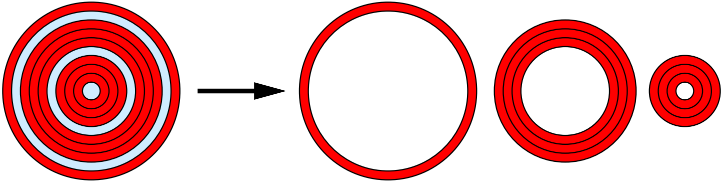
- For a fixed $0 \leq s < k+1$, delete every layer L_i with $i = s \pmod{k+1}$

Baker's shifting strategy for FPT

Subgraph Isomorphism

Input: graphs H and G

Find: a copy of H in G as subgraph.



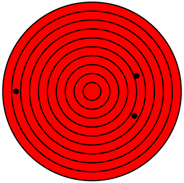
- For a fixed $0 \leq s < k+1$, delete every layer L_i with $i = s \pmod{k+1}$
- The resulting graph is k-outerplanar, hence it has treewidth at most $3k+1$.
- Using the $f(k, \mathrm{tw}) \cdot n$ time algorithm for Subgraph Isomorphism, the problem can be solved in time $f(k, 3k+1) \cdot n$.

Baker's shifting strategy for FPT

SUBGRAPH ISOMORPHISM

Input: graphs H and G

Find: a copy of H in G as subgraph.



We do this for every $0 \leq s < k + 1$:
for at least one value of s, we do not delete
any of the k vertices of the solution

$\Downarrow$

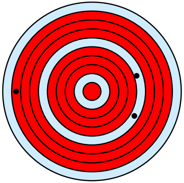
We find a copy of H in G if there is one.

Baker's shifting strategy for FPT

SUBGRAPH ISOMORPHISM

Input: graphs H and G

Find: a copy of H in G as subgraph.



We do this for every $0 \leq s < k+1$:
for at least one value of s, we do not delete
any of the k vertices of the solution

$\Downarrow$

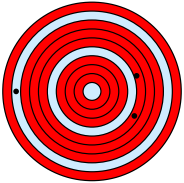
We find a copy of H in G if there is one.

Baker's shifting strategy for FPT

Subgraph Isomorphism

Input: graphs H and G

Find: a copy of H in G as subgraph.



We do this for every $0 \leq s < k+1$:
for at least one value of s, we do not delete
any of the k vertices of the solution

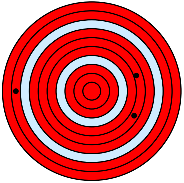
$\Downarrow$

We find a copy of H in G if there is one.

Baker's shifting strategy for FPT

Subgraph Isomorphism

Input: graphs H and G
Find: a copy of H in G as subgraph.



We do this for every $0 \leq s < k+1$:
for at least one value of s, we do not delete
any of the k vertices of the solution

$\Downarrow$

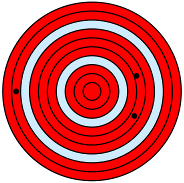
We find a copy of H in G if there is one.

Baker's shifting strategy for FPT

Subgraph Isomorphism

Input: graphs H and G

Find: a copy of H in G as subgraph.



Theorem

Subgraph Isomorphism for planar graphs is FPT parameterized by $k := |V(H)|$.

Baker's shifting strategy for FPT

- The technique is very general, works for many problems on planar graphs:
  - Independent Set
  - Vertex Cover
  - Dominating Set
  - ...
- More generally: First Order Logic problems.
- But for some of these problems, much better techniques are known (see the following slides).

Bidimensionality

A powerful framework for efficient algorithms on planar graphs.

Setup:

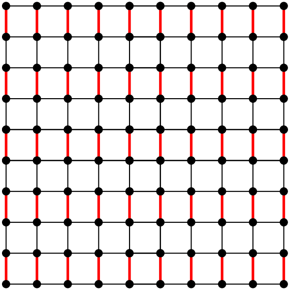
- Let $x(G)$ be some graph invariant (i.e., an integer associated with each graph).
- Given G and k, we want to decide if $x(G) \leq k$ (or $x(G) \geq k$).
- Typical examples:
  - Maximum independent set size.
  - Minimum vertex cover size.
  - Length of the longest path.
  - Minimum dominating set size.
  - Minimum feedback vertex set size.

Bidimensionality [Demaine, Fomin, Hajiaghayi, Thilikos 2005]

For many natural invariants, we can do this in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$ on planar graphs.

Bidimensionality for Vertex Cover

Observation: If the treewidth of a planar graph G is at least $4\sqrt{2k}$
$\Rightarrow$ It has a $\sqrt{2k} \times \sqrt{2k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a matching of size k
$\Rightarrow$ G has a matching of size k
$\Rightarrow$ Vertex cover size is at least k in G.

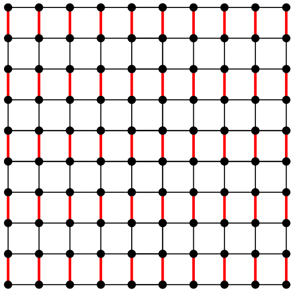


Bidimensionality for Vertex Cover

Observation: If the treewidth of a planar graph G is at least $4\sqrt{2k}$
$\Rightarrow$ It has a $\sqrt{2k} \times \sqrt{2k}$ grid minor (Planar Excluded Grid Theorem)
$\Rightarrow$ The grid has a matching of size k
$\Rightarrow$ G has a matching of size k
$\Rightarrow$ Vertex cover size is at least k in G.

We use this observation to solve Vertex Cover on planar graphs:

- Set $w := 4\sqrt{2k}$.
- Find a 4-approximate tree decomposition.
  - If treewidth is at least w: we answer "vertex cover is $\geq k$."
  - If we get a tree decomposition of width $4w$, then we can solve the problem in time $2^{O(w)} \cdot n^{O(1)} = 2^{O(\sqrt{k})} \cdot n^{O(1)}$.

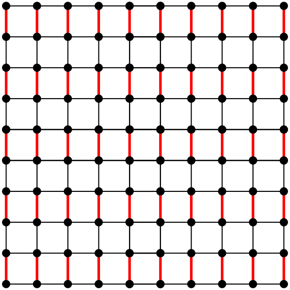


Bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



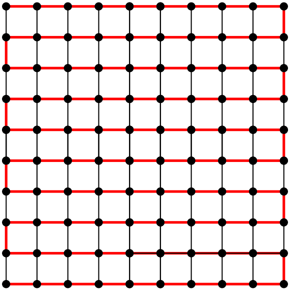
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

Bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



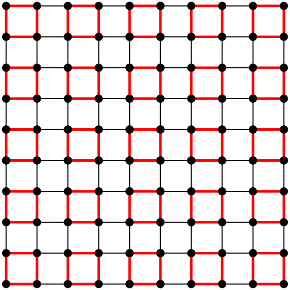
Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

Bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



Examples: minimum vertex cover, length of the longest path, feedback vertex set are minor-bidimensional.

Bidimensionality (cont.)

We can answer "$x(G) \geq k$?" for a minor-bidimensional parameter the following way:

- Set $w := c\sqrt{k}$ for an appropriate constant c.
- Use the 4-approximation tree decomposition algorithm.
  - If treewidth is at least w: $x(G)$ is at least k.
  - If we get a tree decomposition of width $4w$, then we can solve the problem using dynamic programming on the tree decomposition.

Running time:

- If we can solve the problem on tree decomposition of width w in time $2^{O(w)} \cdot n^{O(1)}$, then the running time is $2^{O(\sqrt{k})} \cdot n^{O(1)}$.
- If we can solve the problem on tree decomposition of width w in time $w^{O(w)} \cdot n^{O(1)}$, then the running time is $2^{O(\sqrt{k}\log k)} \cdot n^{O(1)}$.

Contraction bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).

Problem: Dominating Set is **not** minor-bidimensional (why?).

Contraction bidimensionality

Definition

A graph invariant $x(G)$ is **minor-bidimensional** if

- $x(G') \leq x(G)$ for every minor G' of G, and
- If G_k is the $k \times k$ grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).

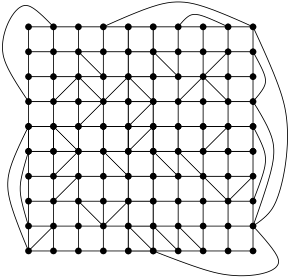
Problem: DOMINATING SET is **not** minor-bidimensional (why?).

We fix the problem by allowing only contractions but not edge/vertex deletions.

Contraction bidimensionality

Theorem

Every **planar graph** with treewidth at least $4k$ can be contracted to a **partially triangulated** $k \times k$ grid.

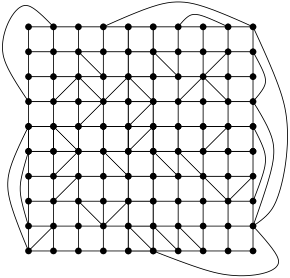


Contraction bidimensionality

Definition

A graph invariant $x(G)$ is **contraction-bidimensional** if

- $x(G') \leq x(G)$ for every contraction G' of G, and
- If G_k is a $k \times k$ partially triangulated grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).

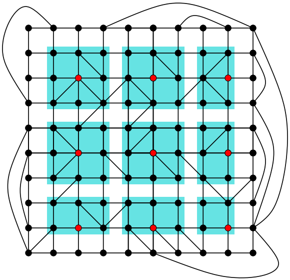


Contraction bidimensionality

Definition

A graph invariant $x(G)$ is **contraction-bidimensional** if

- $x(G') \leq x(G)$ for every contraction G' of G, and
- If G_k is a $k \times k$ partially triangulated grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



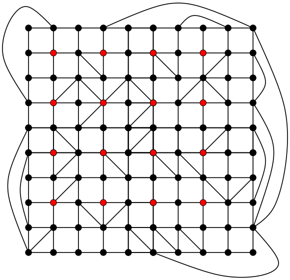
Example: minimum dominating set, maximum independent set are contraction-bidimensional.

Contraction bidimensionality

Definition

A graph invariant $x(G)$ is **contraction-bidimensional** if

- $x(G') \leq x(G)$ for every contraction G' of G, and
- If G_k is a $k \times k$ partially triangulated grid, then $x(G_k) \geq ck^2$ (for some constant $c > 0$).



Example: minimum dominating set, maximum independent set are contraction-bidimensional.

Bidimensionality for Dominating Set

The size of a minimum dominating set is a **contraction bidimensional** invariant: we need at least $(k-2)^2/9$ vertices to dominate all the internal vertices of a partially triangulated $k \times k$ grid (since a vertex can dominate at most 9 internal vertices).

Theorem

Given a tree decomposition of width w, Dominating Set can be solved in time $3^w \cdot w^{O(1)} \cdot n^{O(1)}$.

Solving Dominating Set on planar graphs:

- Set $w := 3\sqrt{k} + 2$.
- Use the 4-approximation tree decomposition algorithm.
  - If treewidth is at least w: we answer 'dominating set is $\geq k$'.
  - If we get a tree decomposition of width $4w$, then we can solve the problem in time $3^w \cdot n^{O(1)} = 2^{O(\sqrt{k})} \cdot n^{O(1)}$.

Constraint Satisfaction Problems (CSP)

A CSP instance is given by describing the

- variables,
- domain of the variables,
- constraints on the variables.

Task: Find an assignment that satisfies every constraint.

$$I = C_1(x_1, x_2, x_3) \wedge C_2(x_2, x_4) \wedge C_3(x_1, x_3, x_4)$$

Constraint Satisfaction Problems (CSP)

A CSP instance is given by describing the

- variables,
- domain of the variables,
- constraints on the variables.

Task: Find an assignment that satisfies every constraint.

$$I = C_1(x_1, x_2, x_3) \wedge C_2(x_2, x_4) \wedge C_3(x_1, x_3, x_4)$$

Examples:

- 3SAT: 2-element domain, every constraint is ternary
- Vertex Coloring: domain is the set of colors, binary constraints
- k-Clique (in graph G): k variables, domain is the vertices of G, $\binom{k}{2}$ binary constraints

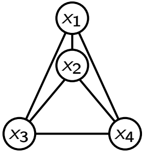
Graphs and hypergraphs related to CSP

Gaifman/primal graph: vertices are the variables, two variables are adjacent if they appear in a common constraint.

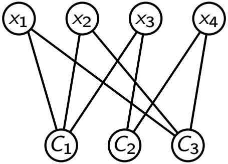
Incidence graph: bipartite graph, vertices are the variables and constraints.

Hypergraph: vertices are the variables, constraints are the hyperedges.

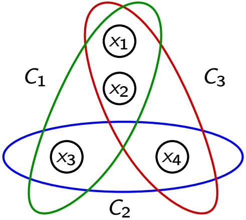
$$I = C_1(x_2, x_1, x_3) \wedge C_2(x_4, x_3) \wedge C_3(x_1, x_4, x_2)$$



Primal graph



Incidence graph



Hypergraph

Treewidth and CSP

Theorem [Freuder 1990]

For every fixed k, CSP can be solved in polynomial time if the primal graph of the instance has treewidth at most k.

Proof sketch:

- Find a tree decomposition of width k (linear-time for fixed k).
- For each bag, enumerate every assignment of the bag that satisfies every constraint fully contained in the bag. Each bag has at most $k+1$ variables, thus there are at most $|D|^{k+1}$ such assignments for each bag.
- Use bottom-up DP to find a satisfying assignment.
- Each constraint induces a clique in the primal graph, thus each constraint is fully contained in one of the bags.
- Running time of DP is polynomial in $|D|^{k+1}$ and the number of variables.

Dichotomy for binary CSP

Binary CSP: Every constraint is of arity 2.

We know that binary CSP($\mathcal{G}$) is polynomial-time solvable for every class $\mathcal{G}$ of graphs with bounded treewidth. Are there other polynomial cases?

Dichotomy for binary CSP

Binary CSP: Every constraint is of arity 2.

We know that binary CSP($\mathcal{G}$) is polynomial-time solvable for every class $\mathcal{G}$ of graphs with bounded treewidth. Are there other polynomial cases?

Theorem [Grohe-Schwentick-Segoufin 2001]

Let $\mathcal{G}$ be a recursively enumerable class of **graphs.** Assuming FPT $\neq$ W[1], the following are equivalent:

- Binary CSP($\mathcal{G}$) is polynomial-time solvable.
- Binary CSP($\mathcal{G}$) is FPT.
- $\mathcal{G}$ has bounded treewidth.

Dichotomy for binary CSP

Binary CSP: Every constraint is of arity 2.

We know that binary CSP($\mathcal{G}$) is polynomial-time solvable for every class $\mathcal{G}$ of graphs with bounded treewidth. Are there other polynomial cases?

Theorem [Grohe-Schwentick-Segoufin 2001]

Let $\mathcal{G}$ be a recursively enumerable class of **graphs.** Assuming FPT $\neq$ W[1], the following are equivalent:

- Binary CSP($\mathcal{G}$) is polynomial-time solvable.
- Binary CSP($\mathcal{G}$) is FPT.
- $\mathcal{G}$ has bounded treewidth.

Note: FPT $\neq$ W[1] is a standard complexity assumption.

Note: Fixed-parameter tractability does not give us more power here than polynomial-time solvability.

Proof outline

Suppose that $\mathcal{G}$ has unbounded treewidth, but CSP($\mathcal{G}$) is FPT.

- Assuming FPT $\neq$ W[1], there is no $f(k)n^c$ time algorithm for k-CLIQUE. But we can solve k-CLIQUE the following way:
- Formulate k-CLIQUE as a binary CSP instance on the $k \times k$ grid.
- Find a $G_k \in \mathcal{G}$ containing a $k \times k$ minor (there is such a G_k by the Excluded Grid Theorem).
- Reduce CSP on the $k \times k$ grid to CSP with graph G_k, which is an instance of CSP($\mathcal{G}$).
- Use the assumed algorithm for CSP($\mathcal{G}$).
- The running time is $f(k)n^c$: the nonpolynomial factors in the running time depend only on k (finding G_k, size of G_k, solving CSP($\mathcal{G}$))
  $\Rightarrow$ k-CLIQUE is FPT, contradicting the hypothesis FPT $\neq$ W[1].

Treewidth — overview

- Algorithms
  - Dynamic programming
  - Courcelle's Theorem
- Properties
  - Characterization by the Cops and Robber game.
  - Excluding a grid, excluding a tree.
  - k-outerplanar graphs.
- Applications
  - Shifting technique for PTAS and FPT.
  - Minor/contraction bidimensionalty.
  - Excluded Grid Theorem in the classification of CSP($\mathcal{G}$).

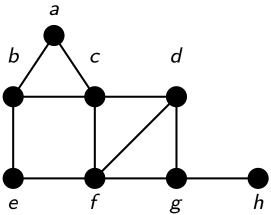
Treewidth

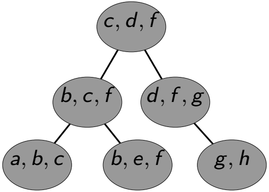
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

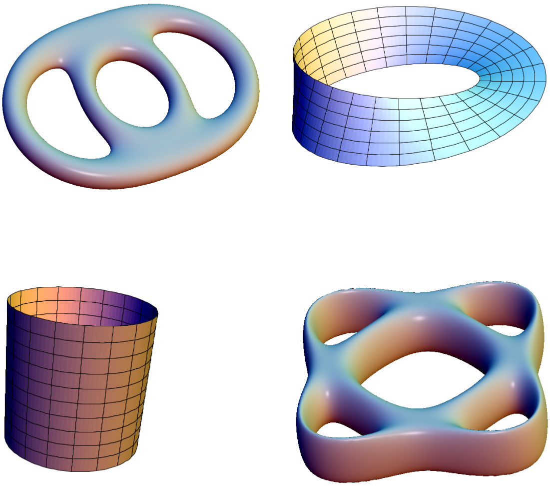
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.







Surfaces

Surfaces

> *A topological surface is a nonempty second countable Hausdorff topological space in which every point has an open neighborhood homeomorphic to some open subset of the Euclidean plane* E^2*.*

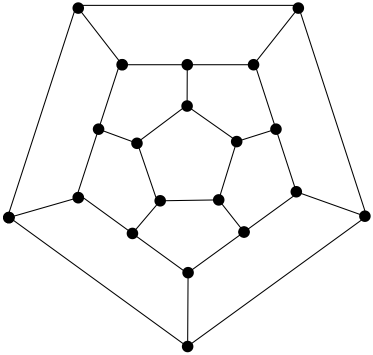
Intuitively: something thin floating in space.

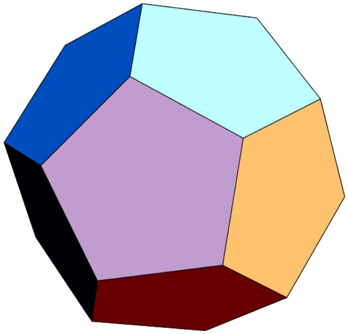
Our viewpoint: which graphs can be drawn on the different surfaces, thus we do not distinguish surfaces that are *homeomorphic.*

Planar graphs

The following are equivalent:

- Graph G can be drawn on the plane.
- Graph G can be drawn inside a disc.
- Graph G can be drawn on the sphere.





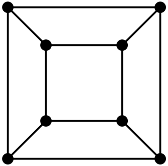
Euler's Formula

Theorem

If G is a connected simple graph drawn in the plane with v vertices, e edges, and f faces, then

$$v + f = e + 2.$$

Example:



$v = 8$
$f = 6$
$e = 12$

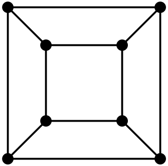
Euler's Formula

Theorem

If G is a connected simple graph drawn in the plane with v vertices, e edges, and f faces, then

$$v + f = e + 2.$$

Example:

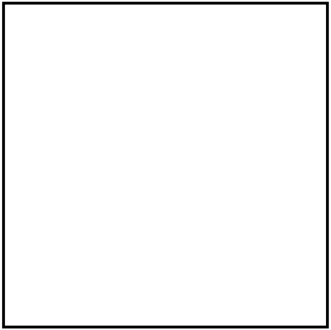


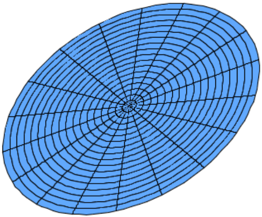
$v = 8$
$f = 6$
$e = 12$

Consequence: $e \leq 3v - 6$

Proof: $2e \geq 3f$ (every face has at least 3 edges)
$e = v + f - 2 \leq v + \frac{2}{3}e - 2$
$\frac{1}{3}e \leq v - 2$
$e \leq 3v - 6$

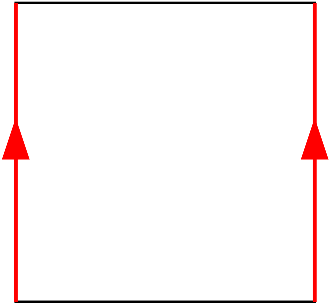
Examples of surfaces: disk

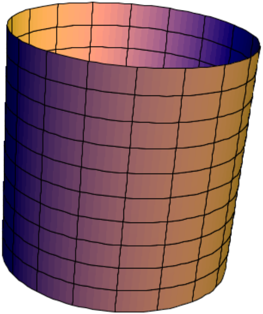




Rectangle with boundary: same as disk.

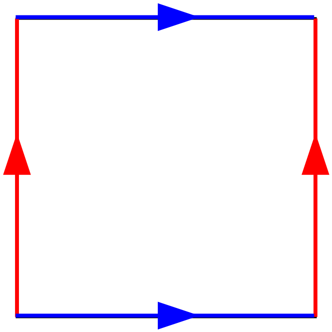
Examples of surfaces: cylinder

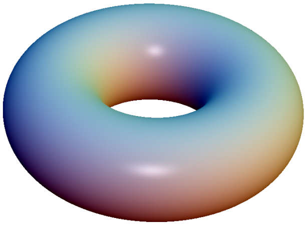




Gluing together the vertical sides creates a cylinder.

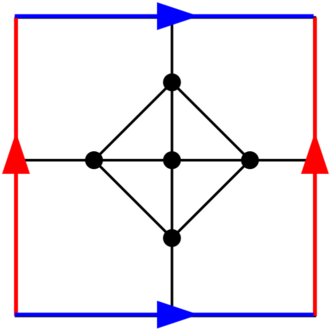
Examples of surfaces: torus

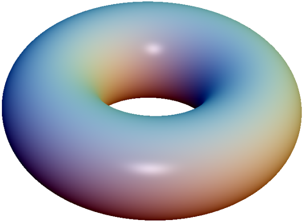




Gluing together both the two horizontal and the two vertical sides creates a torus.

Examples of surfaces: torus



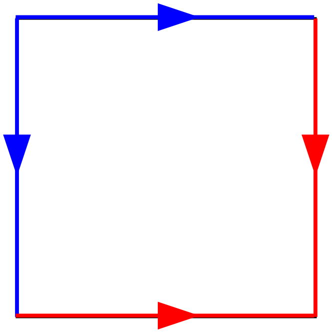


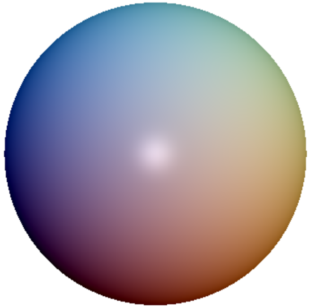
Gluing together both the two horizontal and the two vertical sides creates a torus.

K_5 can be drawn on the torus.

Exercise: draw K_7 on the torus.

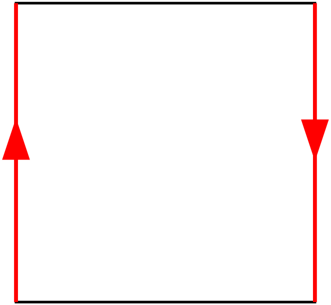
Examples of surfaces: sphere

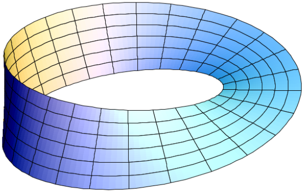




Gluing together top with left and bottom with right creates a sphere.

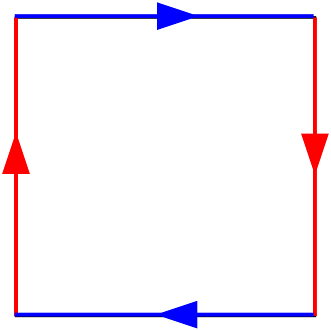
Examples of surfaces: Möbius strip

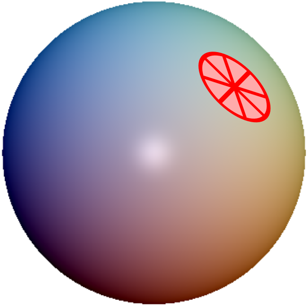




Gluing together the vertical sides in twisted way creates a Möbius strip.

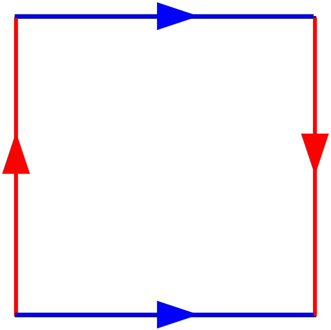
Examples of surfaces: real projective plane





Gluing together both the horizontal and vertical sides in a twisted way creates a real projective plane, which is a sphere with a cross cap.

Examples of surfaces: Klein bottle



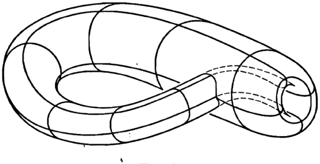
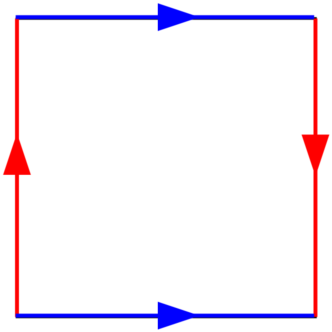
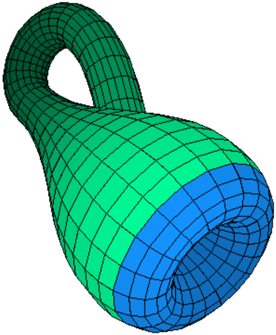


Fig. 297

Gluing together both the horizontal sides in a normal and the vertical sides in a twisted way creates a Klein bottle, which is a sphere with two cross caps.

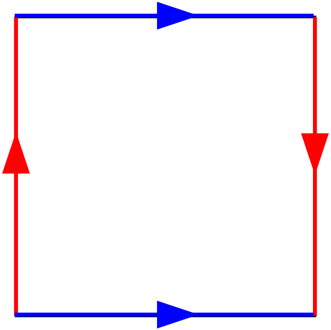
Examples of surfaces: Klein bottle

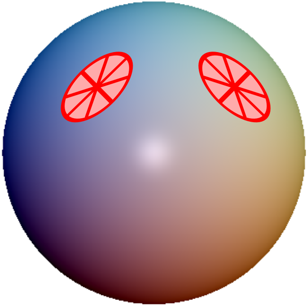




Gluing together both the horizontal sides in a normal and the vertical sides in a twisted way creates a Klein bottle, which is a sphere with two cross caps.

Examples of surfaces: Klein bottle



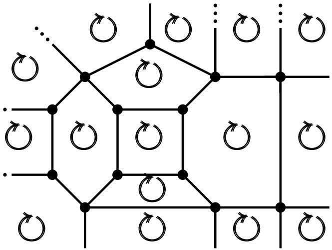


Gluing together both the horizontal sides in a normal and the vertical sides in a twisted way creates a Klein bottle, which is a sphere with two cross caps.

Orientable vs. nonorientable

Definition

A surface Σ is **orientable** if whenever a graph is drawn on Σ such that every face is a disk, then each face can be assigned an orientation such that two faces sharing an edge give the opposite orientation to that edge.



- The sphere and the torus are orientable.
- The Möbius strip and the Klein bottle are nonorientable.

Surfaces with boundaries

Some surfaces have boundaries:

- The cylinder and the Möbius strip have boundaries.
- The sphere, torus, Klein bottle are **closed surfaces.**

Surfaces with boundaries

Some surfaces have boundaries:

- The cylinder and the Möbius strip have boundaries.
- The sphere, torus, Klein bottle are **closed surfaces.**

Every surface with boundaries can be obtained from a closed surface by removing some number of disks.

As removing disks does not change which graphs can be embedded, we consider only closed surfaces from now.

Classification of closed surfaces

Theorem [Brahana 1921]

Every closed surface is equivalent either to

- a sphere with $k \geq 0$ handles (orientable surfaces), or
- or to a sphere with $k \geq 1$ crosscaps (nonorientable surfaces).

Classification of closed surfaces

Theorem [Brahana 1921]

Every closed surface is equivalent either to

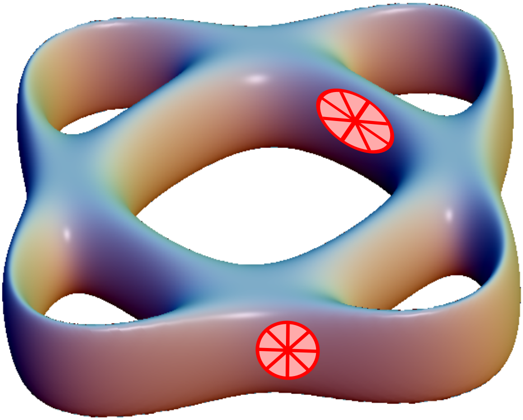
- a sphere with $k \geq 0$ handles (orientable surfaces), or
- or to a sphere with $k \geq 1$ crosscaps (nonorientable surfaces).

Alternative version:

Theorem [Brahana 1921]

Every closed surface is equivalent to a sphere with $k \geq 0$ handles and 0, 1, or 2 crosscaps attached to it.

5 handles, 2 crosscaps



Euler's formula

Theorem

Let G be a connected simple graph drawn on a closed surface Σ such that every face is a disk. If G has v vertices, e edges, and f faces, then

$$v + f = e + 2 - \text{eg}(\Sigma),$$

where the **Euler genus** $\text{eg}(\Sigma)$ is

- $2k$ if Σ is a sphere with k handles, and
- k if Σ is a sphere with k crosscaps.

Consequence: $e \leq 3v - 6 + 3\text{eg}(\Sigma)$

Bounded-genus graphs have bounded average degree.

Algorithms for bounded-genus graphs

Can we generalize the powerful techniques from planar graphs to surfaces?

- Shifting strategy for approximation schemes/parameterized algorithms

  Crucial tool: bounding the treewidth of k-outerplanar graphs.

- Subexponential algorithms for minor/contraction-bidimensional problems.

  Crucial tool: grid theorems.

Grid theorems

Theorem

Every **planar graph** with treewidth at least $4k$ has a $k \times k$ grid minor.

Theorem [Demaine, Fomin, Hajiaghayi, Thilikos 2005]

If G is a graph drawn on Σ and has treewidth at least $c(\mathrm{eg}(\Sigma) + 1) \cdot k$, then G has a $k \times k$ grid minor.

Grid theorems

Theorem

Every **planar graph** with treewidth at least $4k$ has a $k \times k$ grid minor.

Theorem [Demaine, Fomin, Hajiaghayi, Thilikos 2005]

If G is a graph drawn on Σ and has treewidth at least $c(\text{eg}(\Sigma)+1) \cdot k$, then G has a $k \times k$ grid minor.

Subexponential parameterized algorithms for e.g., k-Vertex Cover go through:

- either the graph has a $\Omega(\sqrt{k}) \times \Omega(\sqrt{k})$ grid minor and then there is an independent set of size k, or
- treewidth is $O(\sqrt{k}(\text{eg}(\Sigma)+1))$ and we can solve the problem in time $2^{O(\sqrt{k}(\text{eg}(\Sigma)+1))} \cdot n^{O(1)}$.

Grid theorems

Theorem

Every **planar graph** with treewidth at least $4k$ has a $k \times k$ grid minor.

Theorem [Demaine, Fomin, Hajiaghayi, Thilikos 2005]

If G is a graph drawn on Σ and has treewidth at least $c(\mathrm{eg}(\Sigma)+1) \cdot k$, then G has a $k \times k$ grid minor.

Subexponential parameterized algorithms for e.g., k-Vertex Cover go through:

- either the graph has a $\Omega(\sqrt{k}) \times \Omega(\sqrt{k})$ grid minor and then there is an independent set of size k, or
- treewidth is $O(\sqrt{k}(\mathrm{eg}(\Sigma)+1))$ and we can solve the problem in time $2^{O(\sqrt{k}(\mathrm{eg}(\Sigma)+1))} \cdot n^{O(1)}$.

Similar (more complicated) generalizations for contraction-bidimensional problems.

Local treewidth

The shifting technique relied on the fact that the treewidth of k-outerplanar graphs have bounded treewidth.

Definition

A class $\mathcal{G}$ of graphs has **bounded local treewidth** if there is a function f such that $\mathrm{tw}(G) \leq f(\mathrm{diam}(G))$ for every $G \in \mathcal{G}$.

Bounded genus implies bounded local treewidth:

Theorem

The class $\mathcal{G}_\Sigma$ of graphs embeddable into Σ has bounded local treewidth with $\mathrm{tw}(G) \leq 3\mathrm{eg}(\Sigma)\mathrm{diam}(G)$.

Local treewidth

$B_v[d]$: the **ball** containing vertices at distance $\leq d$ from v.
$R_v[x, y]$ the **ring** containing vertices at distance $x \leq d \leq y$ from v.

Lemma

Let $\mathcal{G}$ be a **minor-closed** class of graphs having **bounded local treewidth.** Then the treewidth of $R_v[x, y]$ can be bounded by a function of $y - x + 1$.

Local treewidth

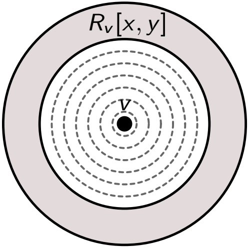
$B_v[d]$: the **ball** containing vertices at distance $\leq d$ from v.
$R_v[x, y]$ the **ring** containing vertices at distance $x \leq d \leq y$ from v.

Lemma

Let $\mathcal{G}$ be a **minor-closed** class of graphs having **bounded local treewidth.** Then the treewidth of $R_v[x, y]$ can be bounded by a function of $y - x + 1$.

Proof:

- Contract $B_v[x-1]$.
- Ring $R_v[x, y]$ appears now at distance $y - x + 1$ from v.
- The ring appears in a graph of treewidth $f(y - x + 1)$.



Local treewidth

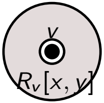
$B_v[d]$: the **ball** containing vertices at distance $\leq d$ from v.
$R_v[x, y]$ the **ring** containing vertices at distance $x \leq d \leq y$ from v.

Lemma

Let $\mathcal{G}$ be a **minor-closed** class of graphs having **bounded local treewidth.** Then the treewidth of $R_v[x, y]$ can be bounded by a function of $y - x + 1$.

Proof:

- Contract $B_v[x-1]$.
- Ring $R_v[x, y]$ appears now at distance $y - x + 1$ from v.
- The ring appears in a graph of treewidth $f(y - x + 1)$.



PTAS using bounded local treewidth

Theorem

If $\mathcal{G}$ is minor-closed and has bounded local treewidth, then INDEPENDENT SET has a PTAS on $\mathcal{G}$.

- Repeat the following for $i = 0, \dots, D$, where $D = \lceil 1/\epsilon \rceil$.
- Pick a vertex v and remove every vertex at distance $jD + i$ for $j = 0, 1, \dots$.
- The graph falls apart into disjoint rings $R_v[0, i-1]$, $R_v[i+1, D+i-1]$, $R_v[D+i+1, 2D+i-1]$, ...
- Thus treewidth is $f(D)$, i.e., can be bounded as function of ϵ.
- Problem can be solved in time $f(1/\epsilon) \cdot n$.
- At least one choice of i removes at most an ϵ fraction of the optimum solution.

Bounded degree

A potential source of confusion

- 3-regular graphs have bounded local treewidth: if diameter is d, then there are at most $\sum_{i=0}^{d} 3^i = (3^{d+1} - 1)/2$ vertices, hence treewidth is bounded by a function of d.
- Independent Set is APX-hard on 3-regular graphs, thus it has no PTAS unless $\mathrm{P} = \mathrm{NP}$.

Have we just proved $\mathrm{P} = \mathrm{NP}$?

Bounded degree

A potential source of confusion

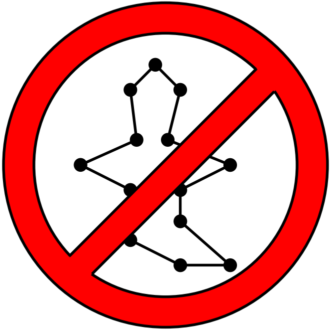
- 3-regular graphs have bounded local treewidth: if diameter is d, then there are at most $\sum_{i=0}^{d} 3^i = (3^{d+1} - 1)/2$ vertices, hence treewidth is bounded by a function of d.
- Independent Set is APX-hard on 3-regular graphs, thus it has no PTAS unless P = NP.

Have we just proved P = NP?

Theorem

If $\mathcal{G}$ is a **minor-closed** and has **bounded local treewidth**, then Independent Set has a PTAS on $\mathcal{G}$.

Local treewidth is useful only for **minor closed** classes!

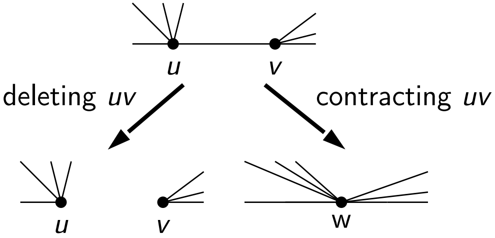


Excluding minors

Minors

Definition

Graph H is a **minor** G ($H \leq G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.

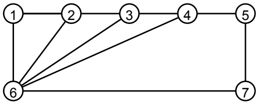


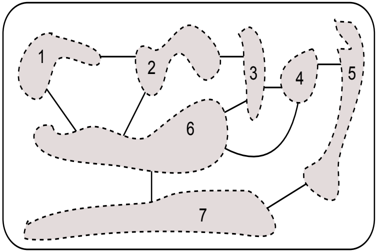
Minors

Equivalent definition

Graph H is a **minor** of G if there is a mapping ϕ (the minor model) that maps each vertex of H to a connected subset of G such that

- $\phi(u)$ and $\phi(v)$ are disjoint if $u \neq v$, and
- if $uv \in E(G)$, then there is an edge between $\phi(u)$ and $\phi(v)$.



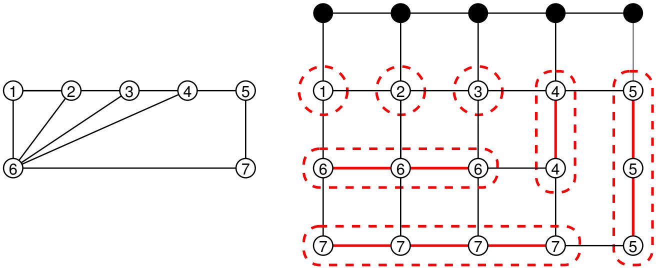


Minors

Equivalent definition

Graph H is a **minor** of G if there is a mapping ϕ (the minor model) that maps each vertex of H to a connected subset of G such that

- $\phi(u)$ and $\phi(v)$ are disjoint if $u \neq v$, and
- if $uv \in E(G)$, then there is an edge between $\phi(u)$ and $\phi(v)$.



Excluding minors

Connection to surfaces:

- Graphs excluding K_5- and $K_{3,3}$-minors are planar.
- Graphs that can be drawn on a fixed surface (e.g., torus) can be characterized by a finite list of excluded minors.

Is it true for every H that H-minor free graphs can be drawn on a fixed surface?

Excluding minors

Connection to surfaces:

- Graphs excluding K_5- and $K_{3,3}$-minors are planar.
- Graphs that can be drawn on a fixed surface (e.g., torus) can be characterized by a finite list of excluded minors.

Is it true for every H that H-minor free graphs can be drawn on a fixed surface?

NO (clique sums), **NO** (apices), **NO** (vortices)

Excluding minors

Connection to surfaces:

- Graphs excluding K_5- and $K_{3,3}$-minors are planar.
- Graphs that can be drawn on a fixed surface (e.g., torus) can be characterized by a finite list of excluded minors.

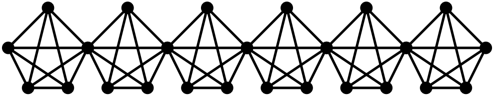
Is it true for every H that H-minor free graphs can be drawn on a fixed surface?

NO (clique sums), **NO** (apices), **NO** (vortices)

YES (in a sense — Robertson-Seymour Structure Theorem)

Excluding minors

The following graph does not have a K_6-minor, but its genus can be large:

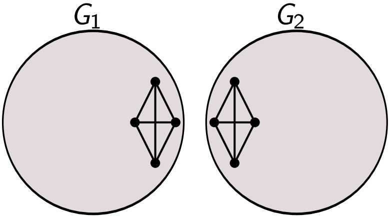


Connecting bounded-genus graphs can increase genus without creating a clique minor.

Clique sums

Definition

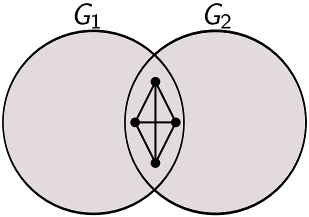
Let G_1 and G_2 be two graphs with two cliques $K_1 \subseteq V(G_1)$ and $K_2 \subseteq V(G_2)$ of the same size. Graph G is a **clique sum** of G_1 and G_2 if it can be obtained by identifying K_1 and K_2, and then removing some of the edges of the clique.



Clique sums

Definition

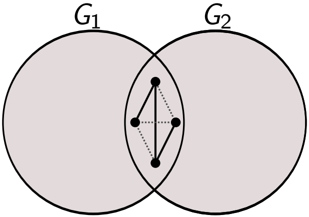
Let G_1 and G_2 be two graphs with two cliques $K_1 \subseteq V(G_1)$ and $K_2 \subseteq V(G_2)$ of the same size. Graph G is a **clique sum** of G_1 and G_2 if it can be obtained by identifying K_1 and K_2, and then removing some of the edges of the clique.



Clique sums

Definition

Let G_1 and G_2 be two graphs with two cliques $K_1 \subseteq V(G_1)$ and $K_2 \subseteq V(G_2)$ of the same size. Graph G is a **clique sum** of G_1 and G_2 if it can be obtained by identifying K_1 and K_2, and then removing some of the edges of the clique.



Clique sums

Observation

If $K_k \not\preceq G_1, G_2$ and G is a clique sum of G_1 and G_2, then $K_k \not\preceq G$.

Thus we can build K_k-minor-free graphs by repeated clique sums.

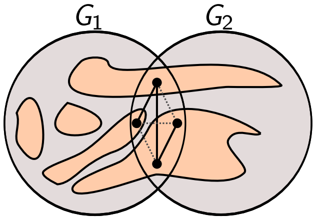
Clique sums

Observation

If $K_k \not\preceq G_1, G_2$ and G is a clique sum of G_1 and G_2, then $K_k \not\preceq G$.

Thus we can build K_k-minor-free graphs by repeated clique sums.

Proof:
For either $i = 1$ or $i = 2$, every set in the model of K_k in G intersects $V(G_i)$. Restricting to $V(G_i)$ gives a model of K_k in G_i (using that the separator is a clique).



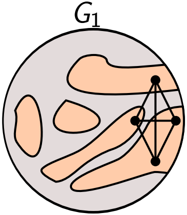
Clique sums

Observation

If $K_k \not\preceq G_1, G_2$ and G is a clique sum of G_1 and G_2, then $K_k \not\preceq G$.

Thus we can build K_k-minor-free graphs by repeated clique sums.

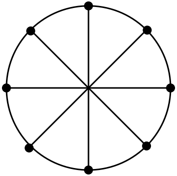
Proof:
For either $i = 1$ or $i = 2$, every set in the model of K_k in G intersects $V(G_i)$. Restricting to $V(G_i)$ gives a model of K_k in G_i (using that the separator is a clique).



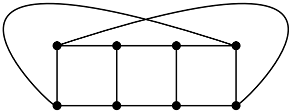
Excluding K_5

Theorem [Wagner 1937]

A graph is K_5-minor-free if and only if it can be built from planar graphs and V_8 by repeated clique sums.



V_8

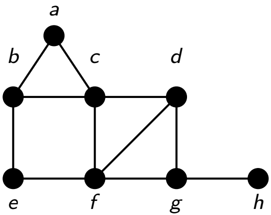


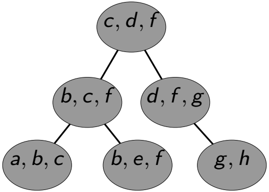
V_8

Tree decomposition

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

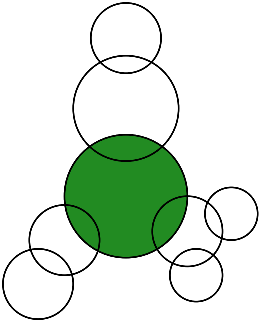
1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.





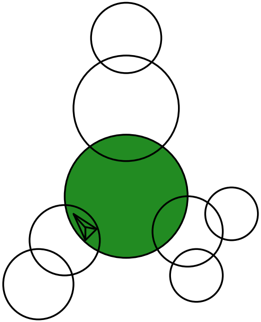
Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



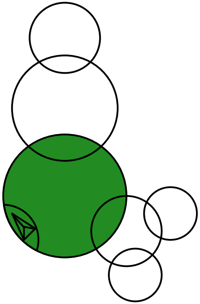
Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



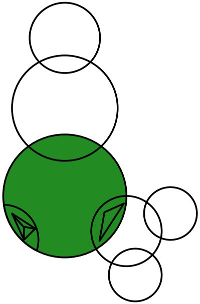
Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



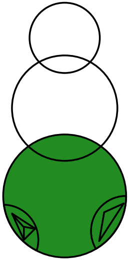
Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



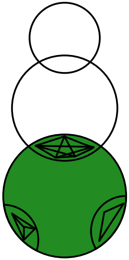
Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



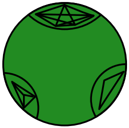
Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



Torso

Torso of a bag: we make the intersections with the adjacent bags cliques.



Excluding K_5 — restated

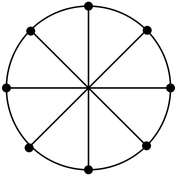
Theorem [Wagner 1937]

A graph is K_5-minor-free if and only if it can be built from planar graphs and V_8 by repeated clique sums.

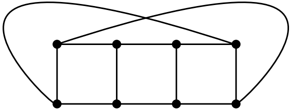
Equivalently:

Theorem [Wagner 1937]

A graph is K_5-minor-free if and only if it has a tree decomposition where every torso is either a planar graph or the graph V_8.



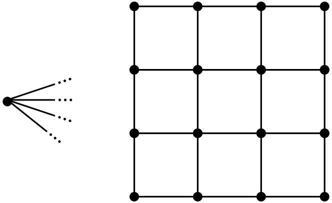
V_8



V_8

Apex vertices

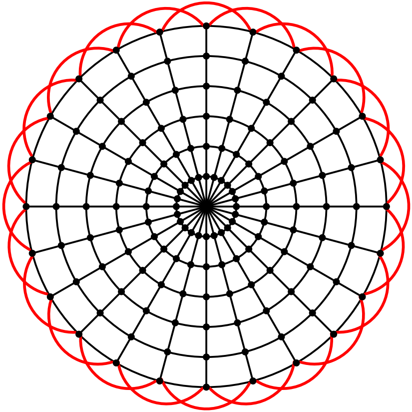
The graph formed from a grid by attaching a universal vertex is K_6-minor-free, but has large genus.



- A planar graph + k extra vertices has no K_{k+5}-minor.
- Instead of bounded genus graphs, our building blocks should be "bounded genus graphs + a bounded number of apex vertices connected arbitrarily."

Vortices

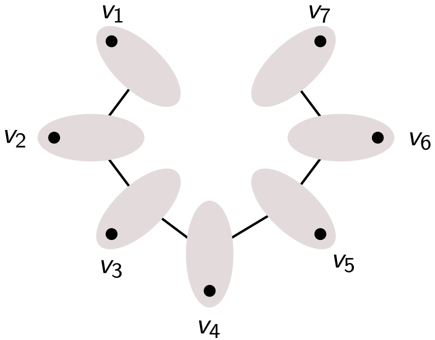
One can show that the following graph has large genus, but cannot have a K_8-minor.



Removing a few apex vertices or decomposing by clique sums does not help.

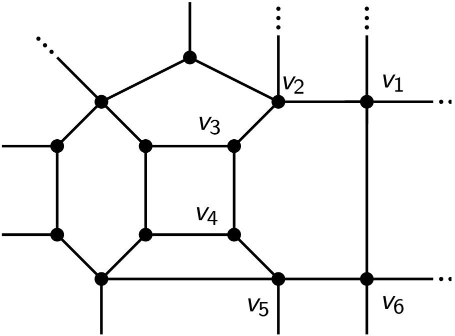
Vortices

- A **vortex** of width k and perimeter $v_1, \ldots, v_n$ is a graph F that has a width-k path decomposition $B_1, \ldots, B_n$ such that $v_i \in B_i$.
- Let G be embedded in Σ and let D be a disk intersecting G only in vertices $v_1, \ldots, v_n$. Attaching a vortex on D means taking the union of G and a vortex on $v_1, \ldots, v_n$ (the vortex intersects G only in these vertices).



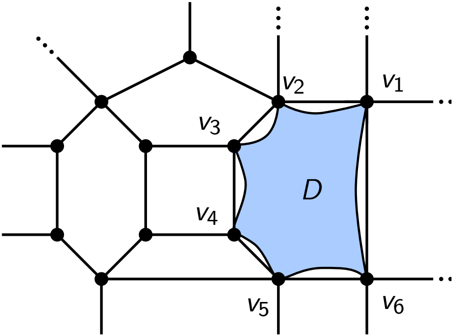
Vortices

- A **vortex** of width k and perimeter $v_1, \dots, v_n$ is a graph F that has a width-k path decomposition $B_1, \dots, B_n$ such that $v_i \in B_i$.
- Let G be embedded in Σ and let D be a disk intersecting G only in vertices $v_1, \dots, v_n$. Attaching a vortex on D means taking the union of G and a vortex on $v_1, \dots, v_n$ (the vortex intersects G only in these vertices).



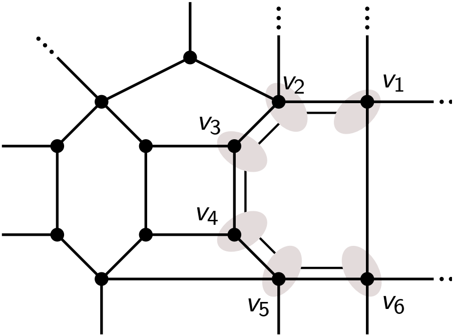
Vortices

- A **vortex** of width k and perimeter $v_1, \ldots, v_n$ is a graph F that has a width-k path decomposition $B_1, \ldots, B_n$ such that $v_i \in B_i$.
- Let G be embedded in Σ and let D be a disk intersecting G only in vertices $v_1, \ldots, v_n$. Attaching a vortex on D means taking the union of G and a vortex on $v_1, \ldots, v_n$ (the vortex intersects G only in these vertices).



Vortices

- A **vortex** of width k and perimeter $v_1, \ldots, v_n$ is a graph F that has a width-k path decomposition $B_1, \ldots, B_n$ such that $v_i \in B_i$.
- Let G be embedded in Σ and let D be a disk intersecting G only in vertices $v_1, \ldots, v_n$. Attaching a vortex on D means taking the union of G and a vortex on $v_1, \ldots, v_n$ (the vortex intersects G only in these vertices).

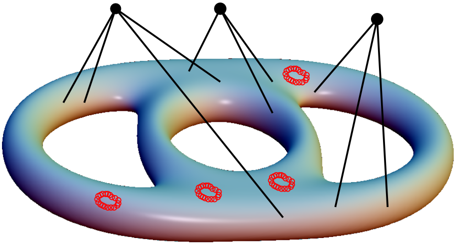


k-almost embeddable

Definition

Graph G is k-**almost embeddable** in surface Σ if

- there is a set X of at most k apex vertices and
- a graph G_0 embedded in Σ, such that
- $G \setminus X$ can be obtained from G_0 by attaching vortices of width k on disjoint disks $D_1, \ldots, D_k$.



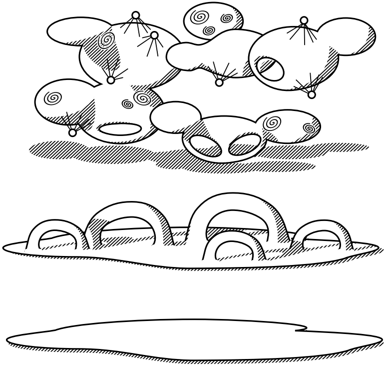
Graph Structure Theorem

Theorem [Robertson-Seymour]

For every graph H, there is an integer k and a surface Σ such that every H-minor-free graph has tree decomposition where every torso is k-almost-embeddable in Σ.

Originally stated only combinatorially, algorithmic versions are known.

- Running time was improved from $n^{f(H)}$ to $f(H) \cdot n^{O(1)}$.
- Algorithm finds also an apex set of size at most k for each torso.



H-Minor-Free

$\cup$

Bounded Genus

$\cup$

Planar

[figure by Felix Reidl]

Excluding cliques

What do we get by excluding small cliques?

- K_3-minor free: every torso is size ≤ 2 (trees).
- K_4-minor free: every torso is size ≤ 3 (series-parallel graphs).
- K_5-minor free: every torso is planar or V_8.
- K_k-minor free for $k \geq 6$: every torso is k-nearly embeddable in some surface.

Algorithmic applications

Theorem [Demaine, Hajiaghayi, Kawarabayashi 2005]

For every graph H, there is a constant c_H such that for any $k \geq 1$, every H-minor-free graph G can be partitioned into $k+1$ vertex sets $V_1, \dots, V_{k+1}$ such that $G \setminus V_i$ has treewidth at most $c_H \cdot k$ for any i. Furthermore, such a partition can be found in polynomial time.

Algorithmic applications

Theorem [Demaine, Hajiaghayi, Kawarabayashi 2005]

For every graph H, there is a constant c_H such that for any $k \geq 1$, every H-minor-free graph G can be partitioned into $k+1$ vertex sets $V_1, \dots, V_{k+1}$ such that $G \setminus V_i$ has treewidth at most $c_H \cdot k$ for any i. Furthermore, such a partition can be found in polynomial time.

PTAS is immediate for e.g., Independent Set:

- Set $k := \lceil 1/\epsilon \rceil$ and find the partition.
- For every $i = 1, \dots, k+1$, compute the solution optimally for $G \setminus V_i$.
- There is one i for which the solution is $k/(k+1) \geq 1 - \epsilon$ times the optimum.

Finding minors

Finding minors

H-minor testing

Input: graph G
Find: a model of H in G.

Finding minors

H-minor testing

Input: graph G

Find: a model of H in G.

Theorem

H-minor testing for **planar** H can be solved in time $f(H) \cdot n^{O(1)}$.

Proof:

- If G has treewidth $\geq g(H)$, then it contains a large grid minor, hence contains H.
- If G has treewidth $< g(H)$, then e.g., Courcelle's Theorem can be invoked to check if G contains an H-minor.

Finding rooted minors

Theorem [Robertson and Seymour]

H-minor testing can be solved in time $f(H) \cdot n^3$.

Robertson and Seymour actually solved a more general problem:

Rooted H-minor testing

Input: graph G, a vertex $\rho(v) \in V(G)$ for every $v \in V(H)$.
Find: a model of H in G where the image of v contains $\rho(v)$.

A very useful special case (let H be a matching with k edges):

k-Disjoint Paths

Input: graph G with vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Find: vertex-disjoint paths $P_1, \ldots, P_k$ where P_i connects s_i and t_i.

Algorithm for minor testing

A vertex $v \in V(G)$ is **irrelevant** if its removal does not change the answer to $H \leq G$.

Ingredients of minor testing by [Robertson and Seymour]

1. Solve the problem on bounded-treewidth graphs.
2. If treewidth is large, either find an **irrelevant** vertex or the model of a large clique minor.
3. If we have a large clique minor, then either we are done (if the clique minor is "close" to the roots), or a vertex of the clique minor is **irrelevant**.

By iteratively removing irrelevant vertices, eventually we arrive to a graph of bounded treewidth.

Planar k-Disjoint Paths

k-Disjoint Paths

Input: graph G with vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Find: vertex-disjoint paths $P_1, \ldots, P_k$ where P_i connects s_i and t_i.

Theorem [Adler et al. 2011]

The k-DISJOINT PATHS problem on planar graphs can be solved in time $2^{2^{O(k)}} \cdot n^{O(1)}$.

Planar k-Disjoint Paths

k-Disjoint Paths

Input: graph G with vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Find: vertex-disjoint paths $P_1, \ldots, P_k$ where P_i connects s_i and t_i.

Theorem [Adler et al. 2011]

The k-Disjoint Paths problem on planar graphs can be solved in time $2^{2^{O(k)}} \cdot n^{O(1)}$.

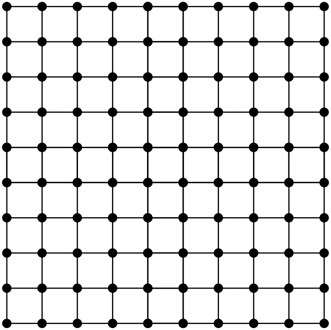
Main argument:

- either treewidth is $2^{O(k)}$ and we can use standard algorithmic techniques of bounded treewidth graphs, or
- treewidth is $2^{\Omega(k)}$ and we can find an **irrelevant vertex** whose deletion does not change the problem.

Planar k-Disjoint Paths

Theorem

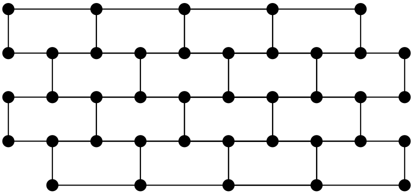
Every planar graph with treewidth at least $4k$ has a $k \times k$ grid minor.



Planar k-Disjoint Paths

Theorem

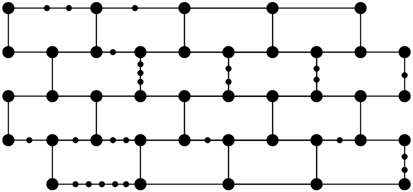
If treewidth of a planar graph is $\Omega(k)$, then it contains the subdivision of a $k \times k$ wall.



Planar k-Disjoint Paths

Theorem

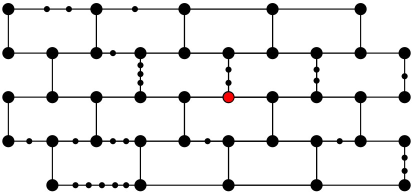
If treewidth of a planar graph is $\Omega(k)$, then it contains the subdivision of a $k \times k$ wall.



Planar k-Disjoint Paths

Theorem

If treewidth of a planar graph is $\Omega(k)$, then it contains the subdivision of a $k \times k$ wall.



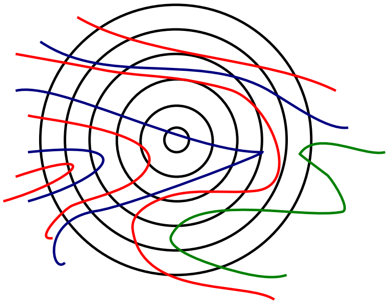
Lemma [Adler et al. 2011]

If a $2^{O(k)} \times 2^{O(k)}$ wall of a planar graph does not enclose any terminals, then the middle vertex of the wall is irrelevant to the k-disjoint paths problem.

Irrelevant vertices

Lemma [Adler et al. 2011]

If there are $2^{O(k)}$ concentric cycles in a planar graph not enclosing any terminals, then the innermost cycle is irrelevant to the k-disjoint paths problem.



Any solution can be rerouted to avoid the innermost cycle.

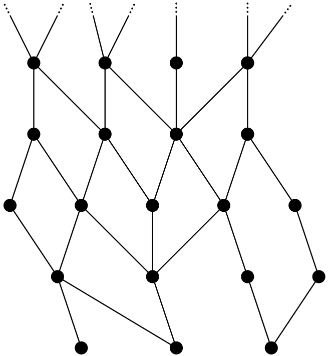
Well-quasi-ordering

Well-quasi-ordering

Definition

A partial order is a **well-quasi-ordering** if

1. There is no infinite antichain.
2. There is no infinite descending chain.

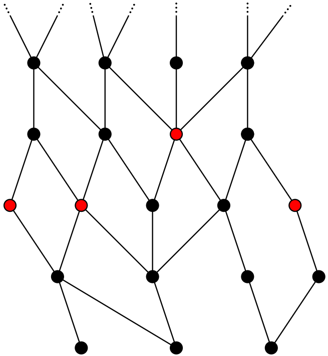


Well-quasi-ordering

Definition

A partial order is a **well-quasi-ordering** if

1. There is no infinite antichain.
2. There is no infinite descending chain.

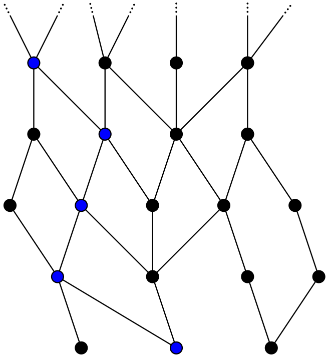


Well-quasi-ordering

Definition

A partial order is a **well-quasi-ordering** if

1. There is no infinite antichain.
2. There is no infinite descending chain.



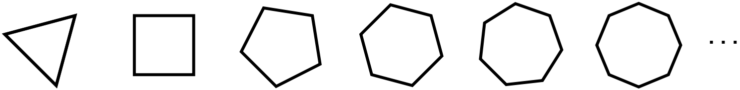
Well-quasi-ordering

Definition

A partial order is a **well-quasi-ordering** if

1. There is no infinite antichain.
2. There is no infinite descending chain.

Example: the subgraph relation $\subseteq$ is **not** a well-quasi-ordering:



Well-quasi-ordering

Graph Minors Theorem

The minor relation $\leq$ is a well-quasi-ordering on finite graphs.

Some equivalent reformulations:

Corollary

If $\mathcal{G}$ is minor closed, then $\overline{\mathcal{G}}$ has a finite number of minimal elements.

Corollary

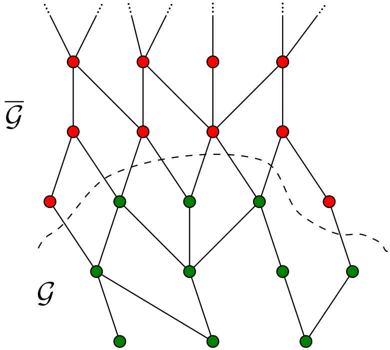
If $\mathcal{G}$ is minor closed, then $\mathcal{G}$ has a finite obstruction set $\mathcal{H} = \{H_1, \ldots, H_k\}$, i.e.,

$$G \in \mathcal{G} \iff \forall H \in \mathcal{F}, H \not\leq G$$

Well-quasi-ordering

Corollary

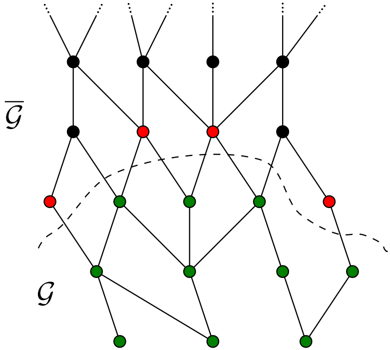
If $\mathcal{G}$ is minor closed, then $\overline{\mathcal{G}}$ has a finite number of minimal elements.



Well-quasi-ordering

Corollary

If $\mathcal{G}$ is minor closed, then $\overline{\mathcal{G}}$ has a finite number of minimal elements.



Nonconstructive algorithms

Corollary

If $\mathcal{G}$ is minor closed, then $\mathcal{G}$ has a finite obstruction set $\mathcal{H} = \{H_1, \ldots, H_k\}$, i.e.,

$$G \in \mathcal{G} \iff \forall H \in \mathcal{F}, H \not\preceq G$$

As we have a $O(n^3)$ minor test algorithm for every $H_i \in \mathcal{H}$:

Theorem

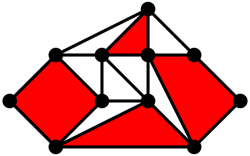
If $\mathcal{G}$ is minor closed, then there is a $O(n^3)$ time algorithm for recognizing graphs in $\mathcal{G}$.

Examples:

- graphs that can be drawn on a torus (double torus etc.) form a minor-closed class: there is a $O(n^3)$ algorithm.
- graphs that have a linkless embedding in 3-space form a minor closed class: there is a $O(n^3)$ algorithm.

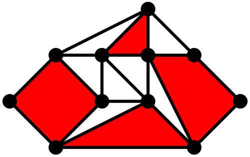
Applications

Planar Face Cover: Given a graph G and an integer k, find an embedding of planar graph G such that there are k faces that cover all the vertices.



Applications

PLANAR FACE COVER: Given a graph G and an integer k, find an embedding of planar graph G such that there are k faces that cover all the vertices.



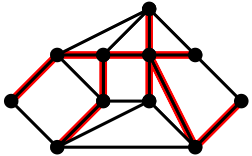
For every fixed k, the class $\mathcal{G}_k$ of graphs of yes-instances is minor closed.

$$\Downarrow$$

For every fixed k, there is a $O(n^3)$ time algorithm for PLANAR FACE COVER.

Applications

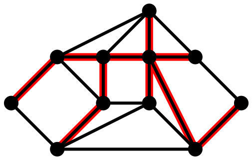
k-LEAF SPANNING TREE: Given a graph G and an integer k, find a spanning tree with **at least** k leaves.



Technical modification: Is there such a spanning tree for at least one component of G?

Applications

k-Leaf Spanning Tree: Given a graph G and an integer k, find a spanning tree with **at least** k leaves.



Technical modification: Is there such a spanning tree for at least one component of G?

For every fixed k, the class $\mathcal{G}_k$ of no-instances is minor closed.

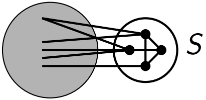
$$\Downarrow$$

For every fixed k, k-Leaf Spanning Tree can be solved in time $O(n^3)$.

$\mathcal{G} + k$ vertices

Definition

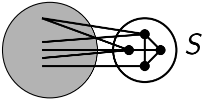
If $\mathcal{G}$ is a graph property, then $\mathcal{G} + kv$ contains graph G if there is a set $S \subseteq V(G)$ of k vertices such that $G \setminus S \in \mathcal{G}$.



$\mathcal{G} + k$ vertices

Definition

If $\mathcal{G}$ is a graph property, then $\mathcal{G} + kv$ contains graph G if there is a set $S \subseteq V(G)$ of k vertices such that $G \setminus S \in \mathcal{G}$.



Observation

If $\mathcal{G}$ is minor closed, then $\mathcal{G} + kv$ is minor closed for every fixed k.

$\Rightarrow$ It is (nonuniform) FPT to decide if G can be transformed into a member of $\mathcal{G}$ by deleting k vertices.

$\mathcal{G} + k$ vertices

Observation

If $\mathcal{G}$ is minor closed, then $\mathcal{G} + kv$ is minor closed for every fixed k.

$\Rightarrow$ It is (nonuniform) FPT to decide if G can be transformed into a member of $\mathcal{G}$ by deleting k vertices.

- If $\mathcal{G}$ = forests $\Rightarrow \mathcal{G} + kv$ = graphs that can be made acyclic by the deletion of k vertices
  $\Rightarrow$ Feedback Vertex Set is FPT.
- If $\mathcal{G}$ = planar graphs $\Rightarrow \mathcal{G} + kv$ = graphs that can be made planar by the deletion of k vertices (k-apex graphs)
  $\Rightarrow$ k-Apex Graph is FPT.
- If $\mathcal{G}$ = empty graphs $\Rightarrow \mathcal{G} + kv$ = graphs with vertex cover number at most k
  $\Rightarrow$ Vertex Cover is FPT.

Nonconstructive algorithms

- The running time is beyond horrible.
- Quick tool for obtaining very general results.
- For many concrete problems, simpler and more efficient algorithms were found.
- Nonuniform FPT: a separate algorithm for every fixed k, rather than a single $f(k) \cdot n^{O(1)}$ algorithm.

What did we learn, Palmer?

- Algorithms for bounded treewidth graphs: tedious, but elementary.
  (dynamic programming, Courcelle's Theorem)
- Applications of bounded treewidth algorithms.
  (the shifting technique, bidimensionality, grid theorems)
- Generalization to bounded genus graphs.
- The structure theorem.
- Minor testing and well-quasi-ordering.