Hitting minors on bounded treewidth graphs

Ignasi Sau

CNRS, LIRMM, Université de Montpellier, France

Joint work with Julien Baste and Dimitrios M. Thilikos
arXiv 1704.07284 + arXiv 1907.04442

1st Joint Meeting Brazil-France in Mathematics
IMPA, Rio de Janeiro, July 2019
Outline of the talk

1. Introduction
   - Parameterized complexity
   - Treewidth
   - FPT algorithms parameterized by treewidth

2. The $\mathcal{F}$-Deletion problem

3. Further research
1 Introduction
   - Parameterized complexity
   - Treewidth
   - FPT algorithms parameterized by treewidth

2 The $\mathcal{F}$-Deletion problem

3 Further research
1 Introduction
- Parameterized complexity
- Treewidth
- FPT algorithms parameterized by treewidth

2 The $\mathcal{F}$-DELETION problem

3 Further research
The area of parameterized complexity

Idea

Measure the complexity of an algorithm in terms of the input size and an additional integer parameter, expected to be small.

This theory started in the late 80’s, by Downey and Fellows:

Today, it is a well-established area with hundreds of articles published every year in the most prestigious TCS journals and conferences.
Parameterized problems

In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is the total input (typically a graph).
- \(k\) is a positive integer called the parameter.

Examples of parameterized problems on graphs, with an instance \((G, k)\):

1. **\(k\)-Vertex Cover**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \leq k\), containing at least an endpoint of every edge?

2. **\(k\)-Clique**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \geq k\), of pairwise adjacent vertices?

3. **Vertex \(k\)-Coloring**: Can \(V(G)\) be colored with \(\leq k\) colors, so that adjacent vertices get different colors?

These three problems are NP-hard, but are they equally hard?
Parameterized problems

In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is the total input (typically a graph).
- \(k\) is a positive integer called the parameter.

Examples of parameterized problems on graphs, with an instance \((G, k)\):

1. \(k\)-Vertex Cover: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \leq k\), containing at least an endpoint of every edge?
2. \(k\)-Clique: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \geq k\), of pairwise adjacent vertices?
3. Vertex \(k\)-Coloring: Can \(V(G)\) be colored with \(|k|\) colors, so that adjacent vertices get different colors?

These three problems are NP-hard, but are they equally hard?
Parameterized problems

In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is the total input (typically a graph).
- \(k\) is a positive integer called the parameter.

Examples of parameterized problems on graphs, with an instance \((G, k)\):

1. **\(k\)-Vertex Cover**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \leq k\), containing at least an endpoint of every edge?
In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is the total input (typically a graph).
- \(k\) is a positive integer called the parameter.

Examples of parameterized problems on graphs, with an instance \((G, k)\):

1. **\(k\)-Vertex Cover**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \leq k\), containing at least an endpoint of every edge?

2. **\(k\)-Clique**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \geq k\), of pairwise adjacent vertices?
Parameterized problems

In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is the total input (typically a graph).
- \(k\) is a positive integer called the parameter.

Examples of parameterized problems on graphs, with an instance \((G, k)\):

1. **k-Vertex Cover**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \leq k\), containing at least an endpoint of every edge?

2. **k-Clique**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \geq k\), of pairwise adjacent vertices?

3. **Vertex k-Coloring**: Can \(V(G)\) be colored with \(\leq k\) colors, so that adjacent vertices get different colors?
Parameterized problems

In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is the total input (typically a graph).
- \(k\) is a positive integer called the parameter.

Examples of parameterized problems on graphs, with an instance \((G, k)\):

1. **\(k\)-Vertex Cover**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \leq k\), containing at least an endpoint of every edge?

2. **\(k\)-Clique**: Does \(G\) contain a set \(S \subseteq V(G)\), with \(|S| \geq k\), of pairwise adjacent vertices?

3. **Vertex \(k\)-Coloring**: Can \(V(G)\) be colored with \(\leq k\) colors, so that adjacent vertices get different colors?

These three problems are NP-hard, but are they equally hard?
They behave quite differently...

1. **$k$-Vertex Cover**: solvable in time $2^k \cdot n^2$

2. **$k$-Clique**: solvable in time $k^2 \cdot n^k$

3. **Vertex $k$-Coloring**: NP-hard for every fixed $k \geq 3$
They behave quite differently...

1. **$k$-Vertex Cover**: solvable in time $2^k \cdot n^2 = f(k) \cdot n^{O(1)}$

2. **$k$-Clique**: solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$

3. **Vertex $k$-Coloring**: NP-hard for every fixed $k \geq 3$
They behave quite differently...

1. **k-Vertex Cover**: solvable in time $2^k \cdot n^2 = f(k) \cdot n^{O(1)}$

   The problem is **FPT** (fixed-parameter tractable)

2. **k-Clique**: solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$

3. **Vertex k-Coloring**: NP-hard for every fixed $k \geq 3$
They behave quite differently...

1. **$k$-Vertex Cover**: solvable in time $2^k \cdot n^2 = f(k) \cdot n^{O(1)}$
   
   The problem is **FPT** (fixed-parameter tractable)

2. **$k$-Clique**: solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$
   
   The problem is **XP** (slice-wise polynomial)

3. **Vertex $k$-Coloring**: NP-hard for every fixed $k \geq 3$
They behave quite differently...

1. **$k$-Vertex Cover**: solvable in time $2^k \cdot n^2 = f(k) \cdot n^{O(1)}$
   - The problem is FPT (fixed-parameter tractable)

2. **$k$-Clique**: solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$
   - The problem is XP (slice-wise polynomial)

3. **Vertex $k$-Coloring**: NP-hard for every fixed $k \geq 3$
   - The problem is para-NP-hard
Introduction
- Parameterized complexity
- Treewidth
- FPT algorithms parameterized by treewidth

The $\mathcal{F}$-Deletion problem

Further research
A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

Example of a 2-tree:

[Figure by Julien Baste]
Treewidth via $k$-trees

A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

Example of a 2-tree:

[Figure by Julien Baste]
A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.
A \textit{k-tree} is a graph that can be built starting from a \((k + 1)\)-clique and then \textit{iteratively} adding a vertex connected to a \textit{k-clique}.

Example of a 2-tree:
A \( k \)-tree is a graph that can be built starting from a \((k + 1)\)-clique and then \textit{iteratively} adding a vertex connected to a \( k \)-clique.

Example of a 2-tree:

[Figure by Julien Baste]
A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

Example of a 2-tree:

[Figure by Julien Baste]
Treewidth via $k$-trees

A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

Example of a 2-tree:
Treewidth via \( k \)-trees

A \( k \)-tree is a graph that can be built starting from a \((k + 1)\)-clique and then iteratively adding a vertex connected to a \( k \)-clique.

Example of a 2-tree:

[Figure by Julien Baste]
A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

Example of a 2-tree:
A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

Example of a 2-tree:

[Figure by Julien Baste]
A $k$-tree is a graph that can be built starting from a $(k + 1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

A partial $k$-tree is a subgraph of a $k$-tree.
A \textit{k-tree} is a graph that can be built starting from a \((k + 1)\)-clique and then \textit{iteratively} adding a vertex connected to a \(k\)-clique.

A partial \textit{k-tree} is a subgraph of a \textit{k-tree}.

\textbf{Treewidth} of a graph \(G\), denoted \(\text{tw}(G)\): smallest integer \(k\) such that \(G\) is a partial \(k\)-tree.
A \textit{k-tree} is a graph that can be built starting from a \((k + 1)\)-clique and then \textit{iteratively} adding a vertex connected to a \textit{k-clique}.

A \textit{partial k-tree} is a subgraph of a \textit{k-tree}.

\textbf{Treewidth} of a graph \(G\), denoted \(\text{tw}(G)\): smallest integer \(k\) such that \(G\) is a partial \(k\)-tree.

Invariant that measures the topological \textit{resemblance} of a graph to a \textit{tree}. 

Example of a 2-tree:
A $k$-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a $k$-clique.

A partial $k$-tree is a subgraph of a $k$-tree.

**Treewidth** of a graph $G$, denoted $\text{tw}(G)$: smallest integer $k$ such that $G$ is a partial $k$-tree.

Invariant that measures the topological resemblance of a graph to a tree.

Construction suggests the notion of tree decomposition: small separators.
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph \( G \):

  pair \( (T, \{B_t \mid t \in V(T)\}) \), where
  - \( T \) is a tree, and
  - \( B_t \subseteq V(G) \ \forall \ t \in V(T) \) (bags),

  satisfying the following:

  - \( \bigcup_{t \in V(T)} B_t = V(G) \),
  - \( \forall \{u, v\} \in E(G), \exists t \in V(T) \) with \( \{u, v\} \subseteq B_t \).
  - \( \forall v \in V(G), \) bags containing \( v \)
    define a connected subtree of \( T \).

- **Width** of a tree decomposition:

  \[ \max_{t \in V(T)} |B_t| - 1. \]

- **Treewidth** of a graph \( G \):

  minimum width of a tree decomposition of \( G \).
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:
  
  pair $(T, \{B_t \mid t \in V(T)\})$, where $T$ is a tree, and $B_t \subseteq V(G)$ $\forall t \in V(T)$ (bags), satisfying the following:

  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  - $\forall\{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width** of a tree decomposition:
  
  $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$:
  
  minimum width of a tree decomposition of $G$. 

An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$: pair $(T, \{B_t \mid t \in V(T)\})$, where $T$ is a tree, and $B_t \subseteq V(G) \forall t \in V(T)$ (bags), satisfying the following:
  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width** of a tree decomposition: $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$: minimum width of a tree decomposition of $G$. 
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:
  
  pair $(T, \{B_t \mid t \in V(T)\})$, where $T$ is a tree, and $B_t \subseteq V(G) \ \forall t \in V(T)$ (bags), satisfying the following:
  
  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  
  - $\forall \{u, v\} \in E(G), \exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width** of a tree decomposition:
  
  $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$:
  
  minimum width of a tree decomposition of $G$. 
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:

  pair $(T, \{B_t \mid t \in V(T)\})$, where $T$ is a tree, and $B_t \subseteq V(G) \ \forall t \in V(T)$ (bags), satisfying the following:

  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  - $\forall \{u, v\} \in E(G), \exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width** of a tree decomposition:
  $$\max_{t \in V(T)} |B_t| - 1.$$

- **Treewidth** of a graph $G$:
  minimum width of a tree decomposition of $G$. 
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:
  
  pair $(T, \{B_t \mid t \in V(T)\})$, where
  
  $T$ is a **tree**, and
  
  $B_t \subseteq V(G) \ \forall t \in V(T)$ (**bags**),

  satisfying the following:

  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  
  - $\forall \{u, v\} \in E(G), \exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  
  - $\forall v \in V(G)$, bags containing $v$ define a **connected** subtree of $T$.

- **Width** of a tree decomposition:
  
  $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$:
  
  minimum width of a tree decomposition of $G$. 

![Diagram of a tree decomposition](image)
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:
  
  pair $(T, \{B_t \mid t \in V(T)\})$, where
  
  $T$ is a tree, and
  
  $B_t \subseteq V(G) \ \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  
  - $\forall \{u, v\} \in E(G), \exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width** of a tree decomposition:
  
  $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$:
  
  minimum width of a tree decomposition of $G$. 
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:
  
  pair $(T, \{B_t \mid t \in V(T)\})$, where $T$ is a tree, and $B_t \subseteq V(G) \ \forall t \in V(T)$ (bags), satisfying the following:
  
  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  - $\forall \{u, v\} \in E(G)$, $\exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width of a tree decomposition**: $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$:
  minimum width of a tree decomposition of $G$. 
An equivalent (and more common) definition of treewidth

- **Tree decomposition** of a graph $G$:
  
  pair $(T, \{B_t \mid t \in V(T)\})$, where
  
  $T$ is a tree, and
  
  $B_t \subseteq V(G) \ \forall t \in V(T)$ (bags),

  satisfying the following:

  - $\bigcup_{t \in V(T)} B_t = V(G)$,
  - $\forall \{u, v\} \in E(G), \exists t \in V(T)$ with $\{u, v\} \subseteq B_t$.
  - $\forall v \in V(G)$, bags containing $v$ define a connected subtree of $T$.

- **Width** of a tree decomposition:
  
  $\max_{t \in V(T)} |B_t| - 1$.

- **Treewidth** of a graph $G$:
  
  minimum width of a tree decomposition of $G$. 
Treewidth is important for (at least) 3 different reasons:
Why treewidth?

Treewidth is important for (at least) 3 different reasons:

1. Treewidth is a fundamental combinatorial tool in graph theory: key role in the Graph Minors project of Robertson and Seymour.
Why treewidth?

Treewidth is important for (at least) 3 different reasons:

1. Treewidth is a fundamental \textit{combinatorial tool} in graph theory: key role in the \textit{Graph Minors} project of Robertson and Seymour.

2. Treewidth behaves very well \textit{algorithmically}, and algorithms parameterized by treewidth appear \textit{very often} in FPT algorithms.
Why treewidth?

Treewidth is important for (at least) 3 different reasons:

1. Treewidth is a fundamental combinatorial tool in graph theory: key role in the Graph Minors project of Robertson and Seymour.

2. Treewidth behaves very well algorithmically, and algorithms parameterized by treewidth appear very often in FPT algorithms.

3. In many practical scenarios, it turns out that the treewidth of the associated graph is small (programming languages, road networks, ...).
Introduction

- Parameterized complexity
- Treewidth
- FPT algorithms parameterized by treewidth

2 The $\mathcal{F}$-DELETION problem

3 Further research
Treewidth behaves very well algorithmically

Monadic Second Order Logic (MSOL): Graph logic that allows quantification over sets of vertices and edges.

Example: $\text{DomSet}(S) : \forall v \in V(G) \exists u \in S : \{u, v\} \in E(G)$

Theorem (Courcelle. 1990) Every problem expressible in MSOL can be solved in time $f(tw) \cdot n$ on graphs on $n$ vertices and treewidth at most $tw$.

In parameterized complexity: FPT parameterized by treewidth.

Examples: Vertex Cover, Dominating Set, Hamiltonian Cycle, Clique, Independent Set, $k$-Coloring for fixed $k$, ...
Monadic Second Order Logic (MSOL):
Graph logic that allows quantification over sets of\textit{ vertices} and \textit{edges}.

\textbf{Example:} \texttt{DomSet}(S): \[ \forall v \in V(G) \setminus S, \exists u \in S : \{u, v\} \in E(G) \]
Treewidth behaves very well algorithmically

Monadic Second Order Logic (MSOL): Graph logic that allows quantification over sets of vertices and edges.

Example: $\text{DomSet}(S) : \left[ \forall v \in V(G) \setminus S, \exists u \in S : \{u, v\} \in E(G) \right]$ 

Theorem (Courcelle. 1990)

Every problem expressible in MSOL can be solved in time $f(tw) \cdot n$ on graphs on $n$ vertices and treewidth at most $tw$.

In parameterized complexity: FPT parameterized by treewidth.
Monadic Second Order Logic (MSOL): Graph logic that allows quantification over sets of vertices and edges.

Example: \( \text{DomSet}(S) : [ \forall v \in V(G) \setminus S, \exists u \in S : \{u, v\} \in E(G) ] \)

Theorem (Courcelle, 1990)

Every problem expressible in MSOL can be solved in time \( f(tw) \cdot n \) on graphs on \( n \) vertices and treewidth at most \( tw \).

In parameterized complexity: FPT parameterized by treewidth.

Examples: Vertex Cover, Dominating Set, Hamiltonian Cycle, Clique, Independent Set, \( k \)-Coloring for fixed \( k \), ...
Only good news?

The vast majority, but not all of them:

List Coloring is $W[1]-$hard parameterized by treewidth. [Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider, Thomassen. 2007]

Some problems involving weights or colors are even NP-hard on graphs of constant treewidth (even on trees!).

For the problems that are FPT parameterized by treewidth, what about the existence of polynomial kernels?

Most natural problems (Vertex Cover, Dominating Set, ...) do not admit polynomial kernels parameterized by treewidth.
1. Are all “natural” graph problems FPT parameterized by treewidth?
Are all “natural” graph problems FPT parameterized by treewidth?

The vast majority, but not all of them:

- **List Coloring** is W[1]-hard parameterized by treewidth.
  
  [Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider, Thomassen. 2007]

- Some problems involving weights or colors are even NP-hard on graphs of constant treewidth (even on trees!).
Are all “natural” graph problems \textbf{FPT parameterized by treewidth}?

The vast \textbf{majority}, but not all of them:

- \textbf{List Coloring} is \textbf{W[1]-hard} parameterized by treewidth.
  
  [Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider, Thomassen. 2007]

- Some problems involving \textbf{weights} or \textbf{colors} are even \textbf{NP-hard} on graphs of \textbf{constant treewidth} (even on trees!).

For the problems that are \textbf{FPT parameterized by treewidth}, what about the existence of \textbf{polynomial kernels}?
Are all “natural” graph problems \textsc{FPT} parameterized by treewidth? 

The vast \textbf{majority}, but not all of them:

- \textbf{List Coloring} is \textit{W[1]}-hard parameterized by treewidth. 
  [Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider, Thomassen. 2007]

- Some problems involving \textit{weights} or \textit{colors} are even \textit{NP-hard} on graphs of \textit{constant treewidth} (even on trees!).

For the problems that are \textbf{FPT} parameterized by \textit{treewidth}, what about the existence of \textit{polynomial kernels}?

Most natural problems (\textit{Vertex Cover, Dominating Set, ...}) do not admit \textit{polynomial kernels} parameterized by \textit{treewidth}.
Typically, Courcelle’s theorem allows to prove that a problem is FPT...

\[ f(tw) \cdot n^{O(1)} \]
Is it enough to prove that a problem is FPT?

Typically, Courcelle’s theorem allows to prove that a problem is FPT...
... but the running time can (and must) be huge!

\[ f(tw) \cdot n^{O(1)} = 2^{345678^{tw}} \cdot n^{O(1)} \]
Is it enough to prove that a problem is FPT?

Typically, Courcelle’s theorem allows to prove that a problem is FPT... ... but the running time can (and must) be huge!

\[ f(tw) \cdot n^{O(1)} = 2^{345678^{tw}} \cdot n^{O(1)} \]

**Major goal** find the smallest possible function \( f(tw) \).

This is a very active area in parameterized complexity.
Typically, Courcelle’s theorem allows to prove that a problem is FPT... ... but the running time can (and must) be huge!

$$f(tw) \cdot n^{O(1)} = 2^{3^{4^{5^{6^{7^{8^{tw}}}}}}} \cdot n^{O(1)}$$

**Major goal** find the smallest possible function $f(tw)$.

This is a very active area in parameterized complexity.

**Remark:** Algorithms parameterized by treewidth appear very often as a “black box” in all kinds of parameterized algorithms.
Is it enough to prove that a problem is FPT?

Typically, Courcelle’s theorem allows to prove that a problem is FPT... ... but the running time can (and must) be huge!

\[ f(tw) \cdot n^{O(1)} = 2^{345678^{tw}} \cdot n^{O(1)} \]

**Major goal** find the **smallest possible** function \( f(tw) \).

This is a very active area in parameterized complexity.

**Remark**: Algorithms parameterized by treewidth appear very often as a “black box” in all kinds of parameterized algorithms.
Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$. 

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

- ETH: The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$.
- SETH: The SAT problem on $n$ variables cannot be solved in time $(2^{1-\epsilon})n$ [Impagliazzo, Paturi. 1999].

SETH $\Rightarrow$ ETH $\Rightarrow$ FPT $\neq W[1]$ $\Rightarrow$ $P \neq NP$.

Typical statements:

- ETH $\Rightarrow$ k-Vertex Cover cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$.
- ETH $\Rightarrow$ Planar k-Vertex Cover cannot in time $2^{o(\sqrt{k})} \cdot n^{O(1)}$. 
Lower bounds on the running times of FPT algorithms

Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$.

Is it possible to obtain an FPT algorithm in time $2^{O(k)} \cdot n^{O(1)}$?

Is it possible to obtain an FPT algorithm in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

- ETH: The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$.
- SETH: The SAT problem on $n$ variables cannot be solved in time $(2^{1-\varepsilon})n$.

[Impagliazzo, Paturi. 1999]

SETH $\Rightarrow$ ETH $\Rightarrow$ FPT $\neq W[1]$ $\Rightarrow$ P $\neq$ NP

Typical statements:

- ETH $\Rightarrow$ $k$-Vertex Cover cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$.
- ETH $\Rightarrow$ Planar $k$-Vertex Cover cannot be solved in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$. 
Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$.

  Is it possible to obtain an FPT algorithm in time $2^{O(k)} \cdot n^{O(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

| ETH: The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$ |
|---------------------------------------------------------------|
| SETH: The SAT problem on $n$ variables cannot be solved in time $(2 - \varepsilon)^n$ |

[Impagliazzo, Paturi. 1999]
Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$.
  
  Is it possible to obtain an FPT algorithm in time $2^{O(k)} \cdot n^{O(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

**ETH:** The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$

**SETH:** The SAT problem on $n$ variables cannot be solved in time $(2 - \varepsilon)^n$

[Impagliazzo, Paturi. 1999]

SETH $\Rightarrow$ ETH
Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$.
  
  Is it possible to obtain an FPT algorithm in time $2^{O(k)} \cdot n^{O(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

**ETH**: The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$

**SETH**: The SAT problem on $n$ variables cannot be solved in time $(2 - \varepsilon)^n$

[Impagliazzo, Paturi. 1999]

SETH $\Rightarrow$ ETH $\Rightarrow$ FPT $\neq$ W[1]
Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{O(k)} \cdot n^{O(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

**ETH**: The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$.

**SETH**: The SAT problem on $n$ variables cannot be solved in time $(2 - \varepsilon)^n$.

[Impagliazzo, Paturi. 1999]

\[ \text{SETH} \Rightarrow \text{ETH} \Rightarrow \text{FPT} \neq \text{W}[1] \Rightarrow P \neq NP \]
Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{O(k)} \cdot n^{O(1)}$.
  
  Is it possible to obtain an FPT algorithm in time $2^{O(k)} \cdot n^{O(1)}$?
  
  Is it possible to obtain an FPT algorithm in time $2^{O(\sqrt{k})} \cdot n^{O(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

ETH: The 3-SAT problem on $n$ variables cannot be solved in time $2^{o(n)}$.

SETH: The SAT problem on $n$ variables cannot be solved in time $(2 - \varepsilon)^n$.

[Impagliazzo, Paturi. 1999]

$$\text{SETH} \Rightarrow \text{ETH} \Rightarrow \text{FPT} \neq \text{W}[1] \Rightarrow \text{P} \neq \text{NP}$$

Typical statements:

ETH $\Rightarrow$ $k$-VERTEX COVER cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$.

ETH $\Rightarrow$ PLANAR $k$-VERTEX COVER cannot be solved in time $2^{o(\sqrt{k})} \cdot n^{O(1)}$. 
Typically, FPT algorithms parameterized by treewidth are based on dynamic programming (DP) over a tree decomposition.
Typically, FPT algorithms parameterized by treewidth are based on dynamic programming (DP) over a tree decomposition.

Starting from the leaves of the tree decomposition, a set of appropriately defined partial solutions is computed recursively until the root, where a global solution is obtained.
Typically, FPT algorithms parameterized by treewidth are based on dynamic programming (DP) over a tree decomposition.

Starting from the leaves of the tree decomposition, a set of appropriately defined partial solutions is computed recursively until the root, where a global solution is obtained.

The way that these partial solutions are defined depends on each particular problem:
Two behaviors for problems parameterized by treewidth

**Local problems**

Vertex Cover, Dominating Set, Clique, Independent Set, \( q \)-Coloring for fixed \( q \).

It is sufficient to store, for each bag \( B \), the subset of vertices of \( B \) that belong to a partial solution: \( O(2^{tw}) \) choices.

The "natural" DP algorithms lead to (optimal) single-exponential algorithms:

\[
2^{O(tw)} \cdot n^{O(1)}.
\]
Two behaviors for problems parameterized by treewidth

Local problems: **Vertex Cover, Dominating Set, Clique, Independent Set, q-Coloring** for fixed $q$. 

It is sufficient to store, for each bag $B$, the subset of vertices of $B$ that belong to a partial solution: $2^{tw} \cdot \Omega(n)$. The "natural" DP algorithms lead to (optimal) single-exponential algorithms: $2^{O(tw)} \cdot n^{O(1)}$. 

$B$ 

18/43
Two behaviors for problems parameterized by treewidth

Local problems: **Vertex Cover, Dominating Set, Clique, Independent Set, q-Coloring** for fixed $q$.

It is sufficient to store, for each bag $B$, the subset of vertices of $B$ that belong to a partial solution: $2^{O(tw)} \cdot n^{O(1)}$. 

18/43
Two behaviors for problems parameterized by treewidth

**Local problems**

\textbf{Vertex Cover}, \textbf{Dominating Set}, \textbf{Clique}, \textbf{Independent Set}, \textbf{q-Coloring} for fixed $q$.

It is sufficient to store, for each bag $B$, the subset of vertices of $B$ that belong to a partial solution: $2^{tw}$ choices.
Two behaviors for problems parameterized by treewidth

**Local problems** VERTEX COVER, DOMINATING SET, CLIQUE, INDEPENDENT SET, $q$-COLORING for fixed $q$.

- It is sufficient to store, for each bag $B$, the subset of vertices of $B$ that belong to a partial solution: $2^{\text{tw}}$ choices.
- The “natural” DP algorithms lead to (optimal) single-exponential algorithms:

$$2^{O(\text{tw})} \cdot n^{O(1)}.$$
Connectivity problems seem to be more complicated...

**Connectivity problems** Hamiltonian Cycle, Longest Path, Steiner Tree, Connected Vertex Cover.

It is not sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number): $2^{O(tw \cdot \log tw)}$ choices.

The "natural" DP algorithms provide only time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$. 

19/43
Connectivity problems seem to be more complicated...

Connectivity problems: \textbf{Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.}

It is not sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number): $2^O(tw \cdot \log tw)$ choices.

The "natural" DP algorithms provide only time $2^O(tw \cdot \log tw) \cdot n^{O(1)}$. 
Connectivity problems seem to be more complicated...

**Connectivity problems** **Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.**
Connectivity problems seem to be more complicated...

Connectivity problems: Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.

It is not sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number):

$$2^{O\left(tw \cdot \log tw\right)}$$

The "natural" DP algorithms provide only time $2^{O\left(tw \cdot \log tw\right)} \cdot n^{O\left(1\right)}$. 

19/43
Connectivity problems seem to be more complicated...

Connectivity problems include: Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.

It is not sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number):

$2^{O\left(tw \cdot \log tw\right)}$ choices

The "natural" DP algorithms provide only time $2^{O\left(tw \cdot \log tw\right)} \cdot n^{O\left(1\right)}$. 
Connectivity problems seem to be more complicated...

Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.

It is not sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number):
Connectivity problems seem to be more complicated...

**Connectivity problems** Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.

- It is **not** sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number):

$$2^{O(tw \cdot \log tw)}$$ choices
Connectivity problems seem to be more complicated...

**Connectivity problems** Hamiltonian Cycle, Longest Cycle, Steiner Tree, Connected Vertex Cover.

It is **not** sufficient to store the subset of vertices of $B$ that belong to a partial solution, but also how they are matched (Bell number):

$$2^{O(tw \cdot \log tw)} \text{ choices}$$

The “natural” DP algorithms provide only time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$. 
There seem to be two behaviors for problems parameterized by treewidth:

- **Local problems:**
  \[2^{O(tw)} \cdot n^{O(1)}\]
  
  *Vertex Cover, Dominating Set, ...*

- **Connectivity problems:**
  \[2^{O(tw \cdot \log tw)} \cdot n^{O(1)}\]
  
  *Longest Path, Steiner Tree, ...*
The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$ were optimal for connectivity problems.
The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$ were optimal for connectivity problems.

This was false!!

Cut&Count technique: [Cygan, Nederlof, Pilipczuk, van Rooij, Wojtaszczyk. 2011]
Randomized single-exponential algorithms for connectivity problems.
The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$ were optimal for connectivity problems.

This was false!!

Cut&Count technique: [Cyg, Nederlof, Pilipczuk, van Rooij, Wojtaszczyk. 2011]

Randomized single-exponential algorithms for connectivity problems.

1. Relax the connectivity requirement by considering a set of cuts that contain the relevant (connected) solutions.
2. Count modulo 2 the number of cuts, because the non-connected solutions will cancel out. By assigning random weights to the vertices/edges, guarantee that w.h.p. the optimal solution is unique (Isolation Lemma).
The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$ were optimal for connectivity problems.

This was false!!

Cut&Count technique: [Cygan, Nederlof, Pilipczuk\textsuperscript{2}, van Rooij, Wojtaszczyk. 2011]

Randomized single-exponential algorithms for connectivity problems.

1. Relax the connectivity requirement by considering a set of cuts that contain the relevant (connected) solutions.

2. Count modulo 2 the number of cuts, because the non-connected solutions will cancel out. By assigning random weights to the vertices/edges, guarantee that w.h.p. the optimal solution is unique (Isolation Lemma).

Deterministic algorithms with algebraic tricks: [Bodlaender, Cygan, Kratsch, Nederlof. 2013]

Representative sets in matroids: [Fomin, Lokshtanov, Saurabh. 2014]
Do all connectivity problems admit single-exponential algorithms (on general graphs) parameterized by treewidth?
End of the story?

Do all connectivity problems admit single-exponential algorithms (on general graphs) parameterized by treewidth?

No!

**Cycle Packing**: find the maximum number of vertex-disjoint cycles.
Do all connectivity problems admit single-exponential algorithms (on general graphs) parameterized by treewidth?

No!

**Cycle Packing**: find the maximum number of vertex-disjoint cycles.

An algorithm in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$ is optimal under the ETH.

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. 2011]

This reduction uses a framework introduced by

[Lokshtanov, Marx, Saurabh. 2011]
Do all connectivity problems admit single-exponential algorithms (on general graphs) parameterized by treewidth? 

No!

**Cycle Packing**: find the maximum number of vertex-disjoint cycles.

An algorithm in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$ is optimal under the ETH.

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. 2011]

This reduction uses a framework introduced by

[Lokshtanov, Marx, Saurabh. 2011]

There are other examples of such problems...
Introduction
- Parameterized complexity
- Treewidth
- FPT algorithms parameterized by treewidth

The $\mathcal{F}$-Deletion problem

Further research
H is a **minor** of a graph G if H can be obtained from a subgraph of G by contracting edges.

*Figure by Gwenaël Joret*
Minors and topological minors

H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges.

H is a topological minor of G if H can be obtained from a subgraph of G by contracting edges with at least one endpoint of deg $\leq 2$.
Minors and topological minors

- $H$ is a **minor** of a graph $G$ if $H$ can be obtained from a subgraph of $G$ by contracting edges.

- $H$ is a **topological minor** of $G$ if $H$ can be obtained from a subgraph of $G$ by contracting edges with at least one endpoint of degree $\leq 2$.

Therefore: $H$ topological minor of $G \Rightarrow H$ minor of $G$
Minors and topological minors

- \( H \) is a **minor** of a graph \( G \) if \( H \) can be obtained from a subgraph of \( G \) by contracting edges.

- \( H \) is a **topological minor** of \( G \) if \( H \) can be obtained from a subgraph of \( G \) by contracting edges with at least one endpoint of \( \text{deg} \leq 2 \).

Therefore: \( H \) topological minor of \( G \) \( \Leftrightarrow \) \( H \) minor of \( G \)
The $\mathcal{F}$-M-Deletion problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.
The $\mathcal{F}$-M-Deletion problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**$\mathcal{F}$-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?
The \texttt{F-M-Deletion} problem

Let $\mathcal{F}$ be a \textbf{fixed finite collection of graphs}.

\begin{tabular}{|l|}
\hline
\textbf{\texttt{F-M-Deletion}} \tabularnewline
\hline
\textbf{Input:} A graph $G$ and an integer $k$. \tabularnewline
\textbf{Parameter:} The treewidth $tw$ of $G$. \tabularnewline
\textbf{Question:} Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a \texttt{minor}? \tabularnewline
\hline
\end{tabular}

- $\mathcal{F} = \{K_2\}$: \texttt{Vertex Cover}.

$\mathcal{F} = \{K_3\}$: \texttt{Feedback Vertex Set}.

"Hardly" solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$.

$\mathcal{F} = \{K_5, K_3, 3\}$: \texttt{Vertex Planarization}.

Solvable in time $2^{\Theta(tw \cdot \log tw)} \cdot n^{O(1)}$.

\cite{Cut&Count. 2011, Jansen, Lokshtanov, Saurabh. 2014 + Pilipczuk. 2015}
The $\mathcal{F}$-M-Deletion problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**F-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: **Vertex Cover**.
  
  Easily solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$. 

The $\mathcal{F}$-M-Deletion problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.

\[ \text{\textbf{F-M-Deletion}} \]

\textbf{Input}: A graph $G$ and an integer $k$.
\textbf{Parameter}: The treewidth $tw$ of $G$.
\textbf{Question}: Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: Vertex Cover.
  Easily solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$.

- $\mathcal{F} = \{K_3\}$: Feedback Vertex Set.
The $\mathcal{F}$-M-Deletion problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**$\mathcal{F}$-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: **Vertex Cover**.
  Easily solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$.

- $\mathcal{F} = \{K_3\}$: **Feedback Vertex Set**.
  “Hardly” solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$.

[Cut&Count. 2011]
The $\mathcal{F}$-**M-Deletion** problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.

$\mathcal{F}$-**M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: **Vertex Cover**.
  Easily solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$.

- $\mathcal{F} = \{K_3\}$: **Feedback Vertex Set**.
  “Hardly” solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$. [Cut&Count. 2011]

- $\mathcal{F} = \{K_5, K_{3,3}\}$: **Vertex Planarization**.
The $\mathcal{F}$-M-Deletion problem

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**$\mathcal{F}$-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $\text{tw}$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: **Vertex Cover**.
  Easily solvable in time $2^{\Theta(\text{tw})} \cdot n^{O(1)}$.

- $\mathcal{F} = \{K_3\}$: **Feedback Vertex Set**.
  “Hardly” solvable in time $2^{\Theta(\text{tw})} \cdot n^{O(1)}$. [Cut&Count. 2011]

- $\mathcal{F} = \{K_5, K_{3,3}\}$: **Vertex Planarization**.
  Solvable in time $2^{\Theta(\text{tw} \cdot \log \text{tw})} \cdot n^{O(1)}$. [Jansen, Lokshtanov, Saurabh. 2014 + Pilipczuk. 2015]
Covering topological minors

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**$\mathcal{F}$-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $\text{tw}$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a minor?

Both problems are NP-hard if $\mathcal{F}$ contains some edge. [Lewis, Yannakakis. 1980]

FPT by Courcelle’s Theorem.
Let $\mathcal{F}$ be a fixed finite collection of graphs.

$\mathcal{F}$-M-Deletion

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a minor?

$\mathcal{F}$-TM-Deletion

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a topol. minor?

Both problems are NP-hard if $\mathcal{F}$ contains some edge. [Lewis, Yannakakis. 1980] FPT by Courcelle's Theorem.
Covering topological minors

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**$\mathcal{F}$-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a minor?

**$\mathcal{F}$-TM-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $tw$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a topol. minor?

Both problems are NP-hard if $\mathcal{F}$ contains some edge. [Lewis, Yannakakis. 1980]
Covering topological minors

Let $\mathcal{F}$ be a fixed finite collection of graphs.

**$\mathcal{F}$-M-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $\text{tw}$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a minor?

**$\mathcal{F}$-TM-Deletion**

**Input:** A graph $G$ and an integer $k$.

**Parameter:** The treewidth $\text{tw}$ of $G$.

**Question:** Does $G$ contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any graph in $\mathcal{F}$ as a topol. minor?

Both problems are NP-hard if $\mathcal{F}$ contains some edge. [Lewis, Yannakakis. 1980]

FPT by Courcelle’s Theorem.
**Objective**

Determine, for every fixed $\mathcal{F}$, the (asymptotically) smallest function $f_{\mathcal{F}}$ such that $\mathcal{F}$-$\text{M-Deletion}$/$\mathcal{F}$-$\text{TM-Deletion}$ can be solved in time $f_{\mathcal{F}}(tw) \cdot n^{O(1)}$ on $n$-vertex graphs.
**Objective**

Determine, for every fixed $\mathcal{F}$, the (asymptotically) smallest function $f_{\mathcal{F}}$ such that $\mathcal{F}$-M-Deletion/$\mathcal{F}$-TM-Deletion can be solved in time

$$f_{\mathcal{F}}(tw) \cdot n^{O(1)}$$

on $n$-vertex graphs.

- We do **not** want to optimize the degree of the polynomial factor.
- We do **not** want to optimize the constants.
- Our hardness results hold under the ETH.
Summary of our results: arXiv 1704.07284+1907.04442

For every $F$:

$F$-M/TM-Deletion in time $2^{O((tw \cdot \log tw) \cdot n^{O(1)})}$.

$F$ connected 1

$F$-M-Deletion in time $2^{O((tw \cdot \log tw) \cdot n^{O(1)})}$.

$F$ planar +

$F$ connected:

$F$-M-Deletion in time $2^{O(tw \cdot n^{O(1)})}$.

(For $F$-TM-Deletion we need: $F$ contains a subcubic planar graph.)

$F$ (connected):

$F$-M/TM-Deletion not in time $2^{o((tw \cdot n^{O(1)})}$ unless the ETH fails, even if $G$ planar.

$F$ = \{H\}, $H$ connected:

complete tight dichotomy.

---

1 Connect\text{ed} collection $\mathcal{F}$: all the graphs are connected.

2 Planar collection $\mathcal{F}$: contains at least one planar graph.
For every $\mathcal{F}$: $\mathcal{F}$-M/TM-Deletion in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$. 

---

1. **Connected** collection $\mathcal{F}$: all the graphs are connected.
2. **Planar** collection $\mathcal{F}$: contains at least one planar graph.
For every $\mathcal{F}$: $\mathcal{F}$-M/TM-Deletion in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.

$\mathcal{F}$ connected$^1$ + planar$^2$: $\mathcal{F}$-M-Deletion in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.

$^1$ Connected collection $\mathcal{F}$: all the graphs are connected.

$^2$ Planar collection $\mathcal{F}$: contains at least one planar graph.
For every \( \mathcal{F} \): \( \mathcal{F} \)-M/TM-\textsc{Deletion} in time \( 2^{O(tw \cdot \log tw)} \cdot n^{O(1)} \).

\( \mathcal{F} \) connected\(^1\) \( \cup \) planar\(^2\): \( \mathcal{F} \)-M-\textsc{Deletion} in time \( 2^{O(tw \cdot \log tw)} \cdot n^{O(1)} \).

\(^1\)Connected collection \( \mathcal{F} \): all the graphs are connected.
\(^2\)Planar collection \( \mathcal{F} \): contains at least one planar graph.
For every $\mathcal{F}$: $\mathcal{F}$-M/TM-Deletion in time $2^{2^{O(tw \cdot \log tw)}} \cdot n^{O(1)}$.  

$\mathcal{F}$ connected$^1$ + planar$^2$: $\mathcal{F}$-M-Deletion in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.  

$G$ planar + $\mathcal{F}$ connected: $\mathcal{F}$-M-Deletion in time $2^{O(tw)} \cdot n^{O(1)}$.  

---

$^1$Connected collection $\mathcal{F}$: all the graphs are connected.  

$^2$Planar collection $\mathcal{F}$: contains at least one planar graph.
For every $\mathcal{F}$: $\mathcal{F}$-M/TM-Deletion in time $2^{2^{O(tw \cdot \log tw)}} \cdot n^{O(1)}$.

$\mathcal{F}$ connected\(^1\) \(\lor\) planar\(^2\): $\mathcal{F}$-M-Deletion in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.

$G$ planar \(\lor\) $\mathcal{F}$ connected: $\mathcal{F}$-M-Deletion in time $2^{O(tw)} \cdot n^{O(1)}$.

(For $\mathcal{F}$-TM-Deletion we need: $\mathcal{F}$ contains a subcubic planar graph.)

\(^1\)Connected collection $\mathcal{F}$: all the graphs are connected.
\(^2\)Planar collection $\mathcal{F}$: contains at least one planar graph.
Summary of our results: arXiv 1704.07284+1907.04442

- For every $\mathcal{F}$: $\mathcal{F}$-M/TM-Deletion in time $2^{O(\text{tw} \cdot \log \text{tw})} \cdot n^{O(1)}$.

- $\mathcal{F}$ connected\textsuperscript{1} or planar\textsuperscript{2}: $\mathcal{F}$-M-Deletion in time $2^{O(\text{tw} \cdot \log \text{tw})} \cdot n^{O(1)}$.

- $G$ planar + $\mathcal{F}$ connected: $\mathcal{F}$-M-Deletion in time $2^{O(\text{tw})} \cdot n^{O(1)}$.

(For $\mathcal{F}$-TM-Deletion we need: $\mathcal{F}$ contains a subcubic planar graph.)

- $\mathcal{F}$ (connected): $\mathcal{F}$-M/TM-Deletion not in time $2^{o(\text{tw})} \cdot n^{O(1)}$ unless the ETH fails, even if $G$ planar.

\textsuperscript{1} Connected collection $\mathcal{F}$: all the graphs are connected.
\textsuperscript{2} Planar collection $\mathcal{F}$: contains at least one planar graph.
Summary of our results: arXiv 1704.07284+1907.04442

- For every $\mathcal{F}$: $\mathcal{F}$-M/TM-Deletion in time $2^{\mathcal{O}(\text{tw} \cdot \log \text{tw})} \cdot n^{\mathcal{O}(1)}$.

- $\mathcal{F}$ connected$^1$ $\pm$ planar$^2$: $\mathcal{F}$-M-Deletion in time $2^{\mathcal{O}(\text{tw} \cdot \log \text{tw})} \cdot n^{\mathcal{O}(1)}$.

- $G$ planar $+$ $\mathcal{F}$ connected: $\mathcal{F}$-M-Deletion in time $2^{\mathcal{O}(\text{tw})} \cdot n^{\mathcal{O}(1)}$.

  (For $\mathcal{F}$-TM-Deletion we need: $\mathcal{F}$ contains a subcubic planar graph.)

- $\mathcal{F}$ (connected): $\mathcal{F}$-M/TM-Deletion not in time $2^{o(\text{tw})} \cdot n^{\mathcal{O}(1)}$ unless the ETH fails, even if $G$ planar.

- $\mathcal{F} = \{H\}$, $H$ connected:

---

1. Connected collection $\mathcal{F}$: all the graphs are connected.
2. Planar collection $\mathcal{F}$: contains at least one planar graph.
For every $\mathcal{F}$: $\mathcal{F}$-\textsc{M/TM-Deletion} in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.

$\mathcal{F}$ connected$^1$ $\oplus$ planar$^2$: $\mathcal{F}$-\textsc{M-Deletion} in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.

$\mathcal{G}$ planar $+$ $\mathcal{F}$ connected: $\mathcal{F}$-\textsc{M-Deletion} in time $2^{O(tw)} \cdot n^{O(1)}$.

(For $\mathcal{F}$-\textsc{TM-Deletion} we need: $\mathcal{F}$ contains a subcubic planar graph.)

$\mathcal{F}$ (connected): $\mathcal{F}$-\textsc{M/TM-Deletion} not in time $2^{o(tw)} \cdot n^{O(1)}$ unless the ETH fails, even if $\mathcal{G}$ planar.

$\mathcal{F} = \{H\}$, $H$ connected: complete tight dichotomy.

---

$^1$ Connected collection $\mathcal{F}$: all the graphs are connected.

$^2$ Planar collection $\mathcal{F}$: contains at least one planar graph.
Complexity of hitting a single connected minor $H$

Classification of the complexity of $\{H\}$-M-Deletion for all connected simple planar graphs $H$ with $|V(H)| \leq 5$ and $|E(H)| \geq 1$: for the 9 graphs on the left (resp. 20 graphs on the right), the problem is solvable in time $2^{\Theta(tw)} \cdot n^{O(1)}$ (resp. $2^{\Theta(tw \cdot \log tw)} \cdot n^{O(1)}$). For $\{H\}$-TM-Deletion, $K_{1,4}$ should be on the left.
For topological minors, there is (at least) one change

$2\Theta(tw)$

$2\Theta(tw \cdot \log tw)$

$P_5$

diamond

$K_4$

$C_5$

$K_{1,4}$

$P_3 \cup 2K_1$

$P_2 \cup P_3$

gem

$K_5$

px

kite

dart

$K_{2,3}$

bull

butterfly

cricket

co-banner
A compact statement for a single connected graph

All these cases can be succinctly described as follows:
A compact statement for a single connected graph

All these cases can be succinctly described as follows:

- All graphs on the left are contractions of \( \square \) or \( \square \).
A compact statement for a single connected graph

All these cases can be succinctly described as follows:

- All graphs on the left are contractions of or
- All graphs on the right are not contractions of or
A dichotomy for hitting connected minors

We can prove that any connected $H$ with $|V(H)| \geq 6$ is \textbf{hard}: \{\text{\textit{H}}\}-\text{\textsc{M-Deletion}} cannot be solved in time $2^{o(tw \cdot \log tw)} \cdot n^{O(1)}$ under the \textbf{ETH}.
A dichotomy for hitting connected minors

We can prove that any connected $H$ with $|V(H)| \geq 6$ is hard: $\{H\}$-$\text{M-Deletion}$ cannot be solved in time $2^{o(tw \cdot \log tw)} \cdot n^{O(1)}$ under the ETH.

**Theorem**

Let $H$ be a connected graph.
We can prove that any connected $H$ with $|V(H)| \geq 6$ is hard: \{H\}-M-Deletion cannot be solved in time $2^{o(tw \cdot \log tw)} \cdot n^{O(1)}$ under the ETH.

**Theorem**

Let $H$ be a connected graph.

The \{H\}-M-Deletion problem is solvable in time

- $2^{O(tw)} \cdot n^{O(1)}$, if $H \preceq_c$ or $H \preceq_c$.
A dichotomy for hitting connected minors

We can prove that any connected $H$ with $|V(H)| \geq 6$ is hard: \{H\}-M-Deletion cannot be solved in time $2^{o(tw \cdot \log tw)} \cdot n^{O(1)}$ under the ETH.

**Theorem**

Let $H$ be a connected graph.

The \{H\}-M-Deletion problem is solvable in time

- $2^{O(tw)} \cdot n^{O(1)}$, if $H \preceq_c \begin{array}{c}
\text{c}\end{array}$ or $H \preceq_c \begin{array}{c}
\text{c}\end{array}$.

- $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$, otherwise.

In both cases, the running time is asymptotically optimal under the ETH.
Why the chair and the banner??

Banner: every connected component (with at least 5 vertices) of a graph that excludes the banner as a (topological) minor is either:
- a cycle (of any length),
- or a tree in which some vertices have been replaced by triangles.

Both such types of components can be maintained by a dynamic programming algorithm in single-exponential time.

If the characterization of the allowed connected components is enriched in some way, such as restricting the length of the allowed cycles or forbidding certain degrees, the problem becomes harder.
Why the chair and the banner??

- **Banner**: every *connected component* (with at least 5 vertices) of a graph that excludes the *banner* as a (topological) minor is either:

  - a cycle (of any length),
  - or a tree in which some vertices have been replaced by triangles.

  Both such types of components can be maintained by a dynamic programming algorithm in single-exponential time.

  If the characterization of the allowed connected components is enriched in some way, such as restricting the length of the allowed cycles or forbidding certain degrees, the problem becomes harder.
Why the chair and the banner??

- **Banner**: every connected component (with at least 5 vertices) of a graph that excludes the banner as a (topological) minor is either:
  - a cycle (of any length),
  - or a tree in which some vertices have been replaced by triangles.
Why the chair and the banner??

- **Banner**: every connected component (with at least 5 vertices) of a graph that excludes the banner as a (topological) minor is either:
  - a cycle (of any length),
  - or a tree in which some vertices have been replaced by triangles.

- Both such types of components can be maintained by a dynamic programming algorithm in single-exponential time.
Why the chair and the banner??

- **Banner**: every connected component (with at least 5 vertices) of a graph that excludes the banner as a (topological) minor is either:
  - a cycle (of any length),
  - or a tree in which some vertices have been replaced by triangles.

- Both such types of components can be maintained by a dynamic programming algorithm in single-exponential time.

- If the characterization of the allowed connected components is enriched in some way, such as restricting the length of the allowed cycles or forbidding certain degrees, the problem becomes harder.
We have three types of results:

1. General algorithms
   - For every $F$: time $O(tw \cdot \log tw) \cdot n^{O(1)}$.
   - $F$ connected + planar: time $O(tw \cdot \log tw) \cdot n^{O(1)}$.
   - $F$ connected + planar: time $O(tw \cdot \log tw) \cdot n^{O(1)}$.
   - $G$ planar + $F$ connected: time $O(tw) \cdot n^{O(1)}$.

2. Ad-hoc single-exponential algorithms
   - Some use “typical” dynamic programming.
   - Some use the rank-based approach.

3. Lower bounds under the ETH
   - $O(tw)$ is “easy”.
   - $O(tw \cdot \log tw)$ is much more involved and we get ideas from:
     - [Lokshtanov, Marx, Saurabh. 2011]
     - [Marcin Pilipczuk. 2017]
     - [Bonnet, Brettell, Kwon, Marx. 2017]
We have three types of results

1. General algorithms

   - For every $F$: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $F$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $F$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $G$ planar + $F$ connected: time $2^{O(tw)} \cdot n^{O(1)}$.

2. Ad-hoc single-exponential algorithms

   - Some use "typical" dynamic programming.
   - Some use the rank-based approach.
   - [Bodlaender, Cygan, Kratsch, Nederlof. 2013]

3. Lower bounds under the ETH

   - $2^{o(tw)}$ is "easy".
   - $2^{o(tw \cdot \log tw)}$ is much more involved and we get ideas from:
     - [Lokshtanov, Marx, Saurabh. 2011]
     - [Marcin Pilipczuk. 2017]
     - [Bonnet, Brettell, Kwon, Marx. 2017]
We have three types of results

1. **General algorithms**
   - For every $\mathcal{F}$: time $2^{2^{O(tw \cdot \log tw)}} \cdot n^{O(1)}$.
   - $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $G$ planar + $\mathcal{F}$ connected: time $2^{O(tw)} \cdot n^{O(1)}$.

2. **Ad-hoc single-exponential algorithms**
   - Some use “typical” dynamic programming.
   - Some use the **rank-based** approach. [Bodlaender, Cygan, Kratsch, Nederlof. 2013]
We have three types of results

1. **General algorithms**
   - For every $\mathcal{F}$: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
   - $G$ planar + $\mathcal{F}$ connected: time $2^{O(tw)} \cdot n^{O(1)}$.

2. **Ad-hoc single-exponential algorithms**
   - Some use “typical” dynamic programming.
   - Some use the rank-based approach. [Bodlaender, Cygan, Kratsch, Nederlof. 2013]

3. **Lower bounds under the ETH**
   - $2^{o(tw)}$ is “easy”.
   - $2^{o(tw \cdot \log tw)}$ is much more involved and we get ideas from:
     - [Lokshtanov, Marx, Saurabh. 2011]
     - [Marcin Pilipczuk. 2017]
     - [Bonnet, Brettell, Kwon, Marx. 2017]
Some ideas of the general algorithms

- For every $\mathcal{F}$: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
- $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
- $G$ planar + $\mathcal{F}$ connected: time $2^{O(tw)} \cdot n^{O(1)}$.
Some ideas of the general algorithms

- For every $\mathcal{F}$: time $2^{2^{O(tw \cdot \log tw)}} \cdot n^{O(1)}$.
- $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
- $G$ planar + $\mathcal{F}$ connected: time $2^{O(tw)} \cdot n^{O(1)}$.

We build on the machinery of **boundaried graphs** and **representatives**:  

[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]  
[Fomin, Lokshtanov, Saurabh, Thilikos. 2010]  
[Kim, Langer, Paul, Reidl, Rossmanith, S., Sikdar. 2013]  
[Garnero, Paul, S., Thilikos. 2014]
Some ideas of the general algorithms

- For every $\mathcal{F}$: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
- $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
- $G$ planar + $\mathcal{F}$ connected: time $2^{O(tw)} \cdot n^{O(1)}$.

We build on the machinery of **boundaried graphs and representatives**:

[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]
[Fomin, Lokshtanov, Saurabh, Thilikos. 2010]
[Kim, Langer, Paul, Reidl, Rossmanith, S., Sikdar. 2013]
[Garnero, Paul, S., Thilikos. 2014]

- $\mathcal{F}$ connected + planar: time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.
  
  **Extra**: Bidimensionality, irrelevant vertices, protrusion decompositions...
Algorithm for a general collection $\mathcal{F}$

- We see $G$ as a $t$-boundaried graph.
Algorithm for a general collection $\mathcal{F}$

- We see $G$ as a $t$-boundaried graph.
- folio of $G$: set of all its $\mathcal{F}$-minor-free minors, up to size $O_\mathcal{F}(t)$. 

For every $t$-boundaried graph $G$,

$$|\text{folio}(G)| = O_\mathcal{F}(1) \cdot (t^2 t) = 2^{O_\mathcal{F}(t \cdot \log t)}.$$ 

This gives an algorithm running in time $2^{2^{O_\mathcal{F}(tw \cdot \log tw)}} \cdot n^{O_\mathcal{F}(1)}$. 

$h$-fold
Algorithm for a general collection $\mathcal{F}$

- We see $G$ as a $t$-boundaried graph.

- folio of $G$: set of all its $\mathcal{F}$-minor-free minors, up to size $\mathcal{O}_\mathcal{F}(t)$.

- We compute, using DP over a tree decomposition of $G$, the following parameter for every folio $C$:

$$ p(G, C) = \min\{|S| : S \subseteq V(G) \land \text{folio}(G - S) = C\} $$
Algorithm for a general collection $\mathcal{F}$

- We see $G$ as a $t$-boundaried graph.

- **folio** of $G$: set of all its $\mathcal{F}$-minor-free minors, up to size $O_\mathcal{F}(t)$.

- We compute, using DP over a tree decomposition of $G$, the following parameter for every folio $C$:

  \[ p(G, C) = \min\{|S| : S \subseteq V(G) \land \text{folio}(G - S) = C\} \]

- For every $t$-boundaried graph $G$,

  \[ |\text{folio}(G)| = O_\mathcal{F}(1) \cdot \binom{t^2}{t} = 2^{O_\mathcal{F}(t \cdot \log t)} \]
Algorithm for a general collection $\mathcal{F}$

- We see $G$ as a $t$-boundaried graph.
- **folio** of $G$: set of all its $\mathcal{F}$-minor-free minors, up to size $O_\mathcal{F}(t)$.

We compute, using DP over a tree decomposition of $G$, the following parameter for every folio $C$:

$$p(G, C) = \min\{|S| : S \subseteq V(G) \land \text{folio}(G - S) = C\}$$

- For every $t$-boundaried graph $G$,
  $$|\text{folio}(G)| = O_\mathcal{F}(1) \cdot \binom{t^2}{t} = 2^{O_\mathcal{F}(t\cdot \log t)}$$

- The number of distinct folios is $2^{2^{O_\mathcal{F}(t\cdot \log t)}}$. 

$G' \subseteq G$
Algorithm for a general collection $\mathcal{F}$

- We see $G$ as a $t$-boundaried graph.
- Folio of $G$: set of all its $\mathcal{F}$-minor-free minors, up to size $\mathcal{O}_\mathcal{F}(t)$.

We compute, using DP over a tree decomposition of $G$, the following parameter for every folio $C$:

$$p(G, C) = \min\{|S| : S \subseteq V(G) \land \text{folio}(G - S) = C\}$$

For every $t$-boundaried graph $G$,

$$|\text{folio}(G)| = \mathcal{O}_{\mathcal{F}}(1) \cdot \binom{t^2}{t} = 2^{\mathcal{O}_{\mathcal{F}}(t \cdot \log t)}$$

The number of distinct folios is $2^{2^{\mathcal{O}_{\mathcal{F}}(t \cdot \log t)}}$.

This gives an algorithm running in time $2^{2^{\mathcal{O}_{\mathcal{F}}(tw \cdot \log tw)}} \cdot n^{\mathcal{O}(1)}$. 

\[skip\]
For a fixed $F$, we define an equivalence relation $\equiv(F, t)$ on $t$-boundary graphs: $G_1 \equiv(F, t) G_2$ if $\forall G' \in B_t, F \preceq m G' \oplus G_1 \iff F \preceq m G' \oplus G_2$.

$R(F, t)$: set of minimum-size representatives of $\equiv(F, t)$.

We compute, using DP over a tree decomposition of $G$, the following parameter for every representative $R$:
\[
p(G, R) = \min \{ |S| : S \subseteq V(G) \land rep_F, t(G - S) = R \}\]

The number of representatives is $|R(F, t)| = 2^{O_F(t \cdot \log t)}$.

This gives an algorithm running in time $2^{O_F(tw \cdot \log tw)} \cdot n^{O(1)}$. 

[Example: Baste, Noy, S. 2017]
Algorithm for a connected and planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv(\mathcal{F}, t)$ on $t$-boundaried graphs:

  $$G_1 \equiv(\mathcal{F}, t) G_2 \quad \text{if } \forall G' \in \mathcal{B}^t,$$

  $$\mathcal{F} \preceq_m G' \oplus G_1 \iff \mathcal{F} \preceq_m G' \oplus G_2.$$
For a fixed \( \mathcal{F} \), we define an equivalence relation \( \equiv^{(\mathcal{F},t)} \) on \( t \)-boundaried graphs:

\[
G_1 \equiv^{(\mathcal{F},t)} G_2 \quad \text{if } \forall G' \in \mathcal{B}^t,
\mathcal{F} \preceq_m G' \oplus G_1 \iff \mathcal{F} \preceq_m G' \oplus G_2.
\]

\( \mathcal{R}^{(\mathcal{F},t)} \): set of minimum-size representatives of \( \equiv^{(\mathcal{F},t)} \).
Algorithm for a connected and planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv(\mathcal{F}, t)$ on $t$-boundaryed graphs:
  $$G_1 \equiv(\mathcal{F}, t) G_2 \quad \text{if } \forall G' \in \mathcal{B}^t, \quad \mathcal{F} \preceq_m G' \oplus G_1 \iff \mathcal{F} \preceq_m G' \oplus G_2.$$  

- $\mathcal{R}(\mathcal{F}, t)$: set of minimum-size representatives of $\equiv(\mathcal{F}, t)$.

- We compute, using DP over a tree decomposition of $G$, the following parameter for every representative $R$:
  $$p(G, R) = \min\{|S| : S \subseteq V(G) \land \text{rep}_{\mathcal{F}, t}(G - S) = R\}$$
Algorithm for a connected and planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv(\mathcal{F},t)$ on $t$-boundaried graphs:
  $$G_1 \equiv(\mathcal{F},t) G_2 \quad \text{if } \forall G' \in B^t, \quad \mathcal{F} \preceq_m G' \oplus G_1 \iff \mathcal{F} \preceq_m G' \oplus G_2.$$  

- $\mathcal{R}(\mathcal{F},t)$: set of minimum-size representatives of $\equiv(\mathcal{F},t)$.

- We compute, using DP over a tree decomposition of $G$, the following parameter for every representative $R$:
  $$p(G, R) = \min\{|S| : S \subseteq V(G) \land \text{rep}_{\mathcal{F},t}(G - S) = R\}$$

- The number of representatives is $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$. 

---

[Baste, Noy, S. 2017] This gives an algorithm running in time $2^{O_{\mathcal{F}}(t \cdot \log t)} \cdot n^{O(1)}$. 

---

#labeled graphs of size $\leq t$ and tw $\leq h$ is $2^{O_h(t \cdot \log t)}$. 

---
Algorithm for a connected and planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv^{(\mathcal{F}, t)}$ on $t$-boundaried graphs:
  \[ G_1 \equiv^{(\mathcal{F}, t)} G_2 \text{ if } \forall G' \in \mathcal{B}^t, \]
  \[ \mathcal{F} \preceq_m G' \oplus G_1 \iff \mathcal{F} \preceq_m G' \oplus G_2. \]

- $\mathcal{R}^{(\mathcal{F}, t)}$: set of minimum-size representatives of $\equiv^{(\mathcal{F}, t)}$.

- We compute, using DP over a tree decomposition of $G$, the following parameter for every representative $R$:
  \[ p(G, R) = \min \{|S| : S \subseteq V(G) \land \text{rep}_{\mathcal{F}, t}(G - S) = R\} \]

- The number of representatives is $|\mathcal{R}^{(\mathcal{F}, t)}| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.
- The number of labeled graphs of size $\leq t$ and $\text{tw} \leq h$ is $2^{O_{h}(t \cdot \log t)}$. [Baste, Noy, S. 2017]
Algorithm for a connected and planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv^{(\mathcal{F},t)}$ on $t$-boundary-restricted graphs:

  $$G_1 \equiv^{(\mathcal{F},t)} G_2 \quad \text{if} \quad \forall G' \in \mathcal{B}^t, \quad \mathcal{F} \preceq_m G' \oplus G_1 \iff \mathcal{F} \preceq_m G' \oplus G_2.$$

- $\mathcal{R}^{(\mathcal{F},t)}$: set of minimum-size representatives of $\equiv^{(\mathcal{F},t)}$.

- We compute, using DP over a tree decomposition of $G$, the following parameter for every representative $R$:

  $$p(G, R) = \min\{|S| : S \subseteq V(G) \land \text{rep}_{\mathcal{F},t}(G - S) = R\}$$

- The number of representatives is $|\mathcal{R}^{(\mathcal{F},t)}| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.

- The number of labeled graphs of size $\leq t$ and $\text{tw} \leq h$ is $2^{O_h(t \cdot \log t)}$. [Baste, Noy, S. 2017]

- This gives an algorithm running in time $2^{O_{\mathcal{F}}(\text{tw} \cdot \log \text{tw})} \cdot n^{O(1)}$. 

---
Algorithm for any connected collection $\mathcal{F}$
Algorithm for any connected collection $\mathcal{F}$
Algorithm for any connected collection $\mathcal{F}$

Embedding with dispersed vertices
[Lemma 15]

Confinement of models inside a railed annulus
[Proposition 13]

Flat Wall Theorem
[12, 32, 44]

Collapse of topological minor models inside a wall
[Theorem 16]

Large $h$-homogeneous subwall
[Lemma 11]

$t \leq \text{tw}(G) + 1$

$h = f(\mathcal{F})$

$R \in \mathcal{R}_{h}^{(i)}$

$R$ contains no irrelevant vertex
[Theorem 19]

$P_{h,r}(R) \leq t$
[Corollary 20]

$|V(R)| = O_h(t)$
[Lemma 25]

Linear protrusion decomposition of $R$
[Lemma 24]

Reduce protrusions [5]
Sparsity of the representatives

$|\mathcal{R}_{h}^{(i)}| = 2^{O_h(t \cdot \log t)}$
[Corollary 27]

[34]

Algorithm in time $O^*(2^{O_h(t \cdot \log tw)})$ for connected $\mathcal{F}$
[Theorem 1]
Hard part: finding an irrelevant vertex inside a flat wall
Hard part: finding an irrelevant vertex inside a flat wall

[Figure by Dimitrios M. Thilikos]
Algorithm when the input graph $G$ is planar

- **Idea** get an improved bound on $|\mathcal{R}(\mathcal{F}, t)|$. 

  We use a sphere-cut decomposition of the input planar graph $G$. 
  
  - Nice topological properties: each separator corresponds to a noose. 
  - The number of representatives is $|\mathcal{R}(\mathcal{F}, t)| = 2^{O(t)}$. 
  - Number of planar triangulations on $t$ vertices is $2^{O(t)}$. 
  - [Tutte. 1962] 

  This gives an algorithm running in time $2^{O(tw)} \cdot n^{O(1)}$. 

  We can extend this algorithm to input graphs $G$ embedded in arbitrary surfaces by using surface-cut decompositions. 

  [Rué, S., Thilikos. 2014]
Algorithm when the input graph $G$ is planar

- **Idea** get an improved bound on $|\mathcal{R}(F,t)|$.

- We use a sphere-cut decomposition of the input planar graph $G$.

  [Seymour, Thomas. 1994] [Dorn, Penninkx, Bodlaender, Fomin. 2010]
Algorithm when the input graph $G$ is planar

- **Idea** get an improved bound on $|\mathcal{R}(\mathcal{F},t)|$.

- We use a sphere-cut decomposition of the input planar graph $G$.

  [Seymour, Thomas. 1994] [Dorn, Penninkx, Bodlaender, Fomin. 2010]

- Nice topological properties: each separator corresponds to a noose.
Algorithm when the input graph $G$ is planar

- **Idea** get an improved bound on $|\mathcal{R}(\mathcal{F}, t)|$.

- We use a **sphere-cut decomposition** of the input **planar graph** $G$.

  [Seymour, Thomas. 1994] [Dorn, Penninkx, Bodlaender, Fomin. 2010]

- **Nice topological properties**: each separator corresponds to a **noose**.

- The number of representatives is $|\mathcal{R}(\mathcal{F}, t)| = 2^{O_\mathcal{F}(t)}$.
  Number of planar triangulations on $t$ vertices is $2^{O(t)}$. [Tutte. 1962]
Algorithm when the input graph $G$ is planar

- **Idea** get an improved bound on $|\mathcal{R}(\mathcal{F},t)|$.

- We use a **sphere-cut decomposition** of the input planar graph $G$.

  [Seymour, Thomas. 1994] [Dorn, Penninkx, Bodlaender, Fomin. 2010]

- Nice topological properties: each separator corresponds to a noose.

- The number of representatives is $|\mathcal{R}(\mathcal{F},t)| = 2^{O(F)(t)}$.
  Number of planar triangulations on $t$ vertices is $2^{O(t)}$.  
  [Tutte. 1962]

- This gives an **algorithm** running in time $2^{O(F)(tw)} \cdot n^{O(1)}$.  

Algorithm when the input graph $G$ is planar

- **Idea** get an improved bound on $|\mathcal{R}(\mathcal{F},t)|$.

- We use a sphere-cut decomposition of the input planar graph $G$.

  [Seymour, Thomas. 1994] [Dorn, Penninkx, Bodlaender, Fomin. 2010]

- Nice topological properties: each separator corresponds to a noose.

- The number of representatives is $|\mathcal{R}(\mathcal{F},t)| = 2^{O(\mathcal{F}(t))}$.

  Number of planar triangulations on $t$ vertices is $2^{O(t)}$. [Tutte. 1962]

- This gives an algorithm running in time $2^{O(\mathcal{F}(tw))} \cdot n^{O(1)}$.

- We can extend this algorithm to input graphs $G$ embedded in arbitrary surfaces by using surface-cut decompositions. [Rué, S., Thilikos. 2014]
Introduction
- Parameterized complexity
- Treewidth
- FPT algorithms parameterized by treewidth

The $\mathcal{F}$-Deletion problem

Further research
What’s next about \( \mathcal{F}\text{-DELETION} \)?

Goal: classify the (asymptotically) tight complexity of \( \mathcal{F}\text{-M-DELETION} \) and \( \mathcal{F}\text{-TM-DELETION} \) for every family \( \mathcal{F} \).

Concerning the minor version: We obtained a tight dichotomy when \(|\mathcal{F}| = 1 \) (connected).

Missing: When \(|\mathcal{F}| \geq 2 \) (connected): \( 2^{\Theta(tw)} \) or \( 2^{\Theta(tw \cdot \log tw)} \)?

Consider families \( \mathcal{F} \) containing disconnected graphs.

Deletion to genus at most \( g \): \( 2^{O(g)(tw \cdot \log tw)} \cdot n^{O(1)} \).

[Kociumaka, Pilipczuk. 2017]

Concerning the topological minor version:

Dichotomy for \( \{H\}\text{-TM-DELETION} \) when \( H \) connected (+planar).

We do not know if there exists some \( \mathcal{F} \) such that \( \mathcal{F}\text{-TM-DELETION} \) cannot be solved in time \( 2^{o(tw^2)} \cdot n^{O(1)} \) under the ETH.

Conjecture: For every (connected) family \( \mathcal{F} \), the \( \mathcal{F}\text{-TM-DELETION} \) problem is solvable in time \( 2^{O(tw \cdot \log tw)} \cdot n^{O(1)} \).
What’s next about $\mathcal{F}$-DELETION?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-M-DELETION and $\mathcal{F}$-TM-DELETION for every family $\mathcal{F}$. 

Concerning the minor version: We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected). Missing: When $|\mathcal{F}| \geq 2$ (connected): $2\Theta(\text{tw})$ or $2\Theta(\text{tw} \cdot \log \text{tw})$?

Consider families $\mathcal{F}$ containing disconnected graphs. Deletion to genus at most $g$: $2O(g)(\text{tw} \cdot \log \text{tw}) \cdot nO(1)$.

[Kociumaka, Pilipczuk. 2017]

Concerning the topological minor version: Dichotomy for $\{H\}$-TM-Deletion when $H$ connected (+planar). We do not know if there exists some $\mathcal{F}$ such that $\mathcal{F}$-TM-DELETION cannot be solved in time $2o(\text{tw}^2) \cdot nO(1)$ under the ETH.

Conjecture For every (connected) family $\mathcal{F}$, the $\mathcal{F}$-TM-DELETION problem is solvable in time $2O(\text{tw} \cdot \log \text{tw}) \cdot nO(1)$. 

42/43
What’s next about $\mathcal{F}$-\textsc{Deletion}?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-\textsc{M-Deletion} and $\mathcal{F}$-\textsc{TM-Deletion} for every family $\mathcal{F}$.

- Concerning the **minor** version:

  - We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected).
  - Missing: When $|\mathcal{F}| \geq 2$ (connected): is $2 \Theta(\text{tw})$ or $2 \Theta(\text{tw} \cdot \log \text{tw})$?
  - Consider families $\mathcal{F}$ containing disconnected graphs.
  - Deletion to genus at most $g$: $2 O(g)(\text{tw} \cdot \log \text{tw}) \cdot n O(1)$.

  - [Kociumaka, Pilipczuk. 2017]

- Concerning the topological minor version:

  - Dichotomy for $\{H\}$-\textsc{TM-Deletion} when $H$ connected (+planar).
  - We do not know if there exists some $\mathcal{F}$ such that $\mathcal{F}$-\textsc{TM-Deletion} cannot be solved in time $2 o(\text{tw}^2) \cdot n O(1)$ under the ETH.

- Conjecture: For every (connected) family $\mathcal{F}$, the $\mathcal{F}$-\textsc{TM-Deletion} problem is solvable in time $2 O(\text{tw} \cdot \log \text{tw}) \cdot n O(1)$. 
What’s next about $\mathcal{F}$-Deletion?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-M-Deletion and $\mathcal{F}$-TM-Deletion for every family $\mathcal{F}$.

- Concerning the **minor** version:
  - We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected).

- Consider families $\mathcal{F}$ containing disconnected graphs.
  - Deletion to genus at most $g$:
    \[ 2O_g\left(\tw \cdot \log \tw\right)\cdot n^{O(1)} \]
    \cite{Kociumaka, Pilipczuk. 2017}

- Concerning the topological minor version:
  - Dichotomy for $\{H\}$-TM-Deletion when $H$ connected (+planar).
  - We do not know if there exists some $\mathcal{F}$ such that $\mathcal{F}$-TM-Deletion cannot be solved in time $2^{o\left(\tw^2\right)} \cdot n^{O(1)}$ under the ETH.

- **Conjecture** For every (connected) family $\mathcal{F}$, the $\mathcal{F}$-TM-Deletion problem is solvable in time $2^{O\left(\tw \cdot \log \tw\right)} \cdot n^{O(1)}$. 

42/43
What's next about $\mathcal{F}$-Deletion?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-M-Deletion and $\mathcal{F}$-TM-Deletion for every family $\mathcal{F}$.

- Concerning the **minor** version:
  - We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected).
  - **Missing**: When $|\mathcal{F}| \geq 2$ (connected): $2^{\Theta(tw)}$ or $2^{\Theta(tw \cdot \log tw)}$?
What’s next about $\mathcal{F}$-DELETION?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-M-DELETION and $\mathcal{F}$-TM-DELETION for every family $\mathcal{F}$.

- Concerning the **minor** version:
  - We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected).
  - **Missing:** When $|\mathcal{F}| \geq 2$ (connected): $2^{\Theta(tw)}$ or $2^{\Theta(tw \cdot \log tw)}$?
  - Consider families $\mathcal{F}$ containing disconnected graphs.
What’s next about \( \mathcal{F} \text{-Deletion} \)?

- **Goal** classify the (asymptotically) tight complexity of \( \mathcal{F} \text{-M-Deletion} \) and \( \mathcal{F} \text{-TM-Deletion} \) for every family \( \mathcal{F} \).

- Concerning the **minor** version:
  - We obtained a tight dichotomy when \( |\mathcal{F}| = 1 \) (connected).
  - **Missing:** When \( |\mathcal{F}| \geq 2 \) (connected): \( 2^{\Theta(tw)} \) or \( 2^{\Theta(tw \cdot \log tw)} \)?
  - Consider families \( \mathcal{F} \) containing **disconnected graphs**.
    Deletion to **genus at most** \( g \): \( 2^{O_g(tw \cdot \log tw)} \cdot n^{O(1)} \). [Kociumaka, Pilipczuk. 2017]
What’s next about $\mathcal{F}$-\textsc{Deletion}?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-\textsc{M-Deletion} and $\mathcal{F}$-\textsc{TM-Deletion} for every family $\mathcal{F}$.

Concerning the **minor** version:

- We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected).

- **Missing**: When $|\mathcal{F}| \geq 2$ (connected): $2^{\Theta(tw)}$ or $2^{\Theta(tw \cdot \log tw)}$?

- Consider families $\mathcal{F}$ containing disconnected graphs. Deletion to genus at most $g$: $2^{O_g(tw \cdot \log tw)} \cdot n^{O(1)}$. [Kociumaka, Pilipczuk. 2017]

Concerning the **topological minor** version:
What’s next about $\mathcal{F}$-Deletion?

- **Goal** classify the (asymptotically) tight complexity of $\mathcal{F}$-M-Deletion and $\mathcal{F}$-TM-Deletion for every family $\mathcal{F}$.

Concerning the **minor** version:

- We obtained a tight dichotomy when $|\mathcal{F}| = 1$ (connected).
- **Missing:** When $|\mathcal{F}| \geq 2$ (connected): $2^{\Theta(tw)}$ or $2^{\Theta(tw \cdot \log tw)}$?

- Consider families $\mathcal{F}$ containing disconnected graphs.
  Deletion to genus at most $g$: $2^{O_g(tw \cdot \log tw)} \cdot n^{O(1)}$. [Kociumaka, Pilipczuk. 2017]

Concerning the **topological minor** version:

- Dichotomy for $\{H\}$-TM-Deletion when $H$ connected (+planar).
What’s next about \( \mathcal{F}\)-Deletion?

- **Goal** classify the (asymptotically) tight complexity of \( \mathcal{F}\)-M-Deletion and \( \mathcal{F}\)-TM-Deletion for every family \( \mathcal{F} \).

Concerning the **minor** version:
- We obtained a tight dichotomy when \(|\mathcal{F}| = 1\) (connected).
- **Missing:** When \(|\mathcal{F}| \geq 2\) (connected): \(2^{\Theta(tw)}\) or \(2^{\Theta(tw \cdot \log tw)}\)?
- Consider families \( \mathcal{F} \) containing disconnected graphs.
  Deletion to genus at most \( g \): \(2^{O_g(tw \cdot \log tw)} \cdot n^{O(1)}\). [Kociumaka, Pilipczuk. 2017]

Concerning the **topological minor** version:
- Dichotomy for \( \{H\}\)-TM-Deletion when \( H \) connected (+planar).
- We do not know if there exists some \( \mathcal{F} \) such that \( \mathcal{F}\)-TM-Deletion cannot be solved in time \(2^{o(tw^2)} \cdot n^{O(1)}\) under the ETH.
What’s next about \textit{F-Deletion}?

- **Goal** classify the (asymptotically) tight complexity of \textit{F-M-Deletion} and \textit{F-TM-Deletion} for every family \(F\).

- Concerning the **minor** version:
  - We obtained a tight dichotomy when \(|F| = 1\) (connected).
  - **Missing:** When \(|F| \geq 2\) (connected): \(2^{\Theta(tw)}\) or \(2^{\Theta(tw \cdot \log tw)}\)?
  - Consider families \(F\) containing disconnected graphs. Deletion to genus at most \(g\): \(2^{O_g(tw \cdot \log tw)} \cdot n^{O(1)}\). [Kociumaka, Pilipczuk. 2017]

- Concerning the **topological minor** version:
  - Dichotomy for \(\{H\}\)-TM-Deletion when \(H\) connected (+planar).
  - We do not know if there exists some \(F\) such that \(F\)-TM-Deletion cannot be solved in time \(2^{o(tw^2)} \cdot n^{O(1)}\) under the ETH.
  - **Conjecture** For every (connected) family \(F\), the \(F\)-TM-Deletion problem is solvable in time \(2^{O(tw \cdot \log tw)} \cdot n^{O(1)}\).
Gràcies!