Efficient algorithms parameterized by treewidth

Ignasi Sau

CNRS, LIRMM, Université de Montpellier, France

Algorithms and Combinatorics seminar
PESC-UFRJ, Rio de Janeiro, Brazil
December 2019
Outline of the talk

1 Introduction
   - Parameterized complexity
   - Treewidth

2 FPT algorithms parameterized by treewidth

3 The $\mathcal{F}$-Deletion problem
Introduction

- Parameterized complexity
- Treewidth

FPT algorithms parameterized by treewidth

The $\mathcal{F}$-Deletion problem
1 Introduction
   • Parameterized complexity
   • Treewidth

2 FPT algorithms parameterized by treewidth

3 The \( \mathcal{F} \)-Deletion problem
Crucial notion in complexity theory: NP-completeness

- Cook-Levin Theorem (1971): the SAT problem is NP-complete.

- Karp (1972): list of 21 important NP-complete problems.

- Nowadays, literally thousands of problems are known to be NP-hard: unless $P = NP$, they cannot be solved in polynomial time.
Crucial notion in complexity theory: NP-completeness

- Cook-Levin Theorem (1971): the SAT problem is NP-complete.

- Karp (1972): list of 21 important NP-complete problems.

- Nowadays, literally thousands of problems are known to be NP-hard: unless $P = NP$, they cannot be solved in polynomial time.

- But what does it mean for a problem to be NP-hard?

  No algorithm solves all instances optimally in polynomial time.
Are all instances really hard to solve?

Maybe there are relevant subsets of instances that can be solved efficiently.
Are all instances really hard to solve?

Maybe there are relevant subsets of instances that can be solved efficiently.

- **VLSI design**: the number of circuit layers is usually $\leq 10$.
- **Computational biology**: Real instances of DNA chain reconstruction usually have treewidth $\leq 11$.
- **Robotics**: Number of degrees of freedom in motion planning problems $\leq 10$.
- **Compilers**: Checking compatibility of type declarations is hard, but usually the depth of type declarations is $\leq 10$. 
Are all instances really hard to solve?

Maybe there are relevant subsets of instances that can be solved efficiently.

- **VLSI design**: the number of circuit layers is usually $\leq 10$.
- **Computational biology**: Real instances of DNA chain reconstruction usually have treewidth $\leq 11$.
- **Robotics**: Number of degrees of freedom in motion planning problems $\leq 10$.
- **Compilers**: Checking compatibility of type declarations is hard, but usually the depth of type declarations is $\leq 10$.

**Message** In many applications, not only the total size of the instance matters, but also the value of an additional parameter.
The area of parameterized complexity

Idea
Measure the complexity of an algorithm in terms of the input size and an additional integer parameter.

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

- \(x\) is a typical input (in our setting, 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?
In a parameterized problem, an instance is a pair \((x, k)\), where

- \(x\) is a typical input (in our setting, 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 a typical input (in our setting, 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?
Parameterized problems

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

- \(x\) is a typical input (in our setting, 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 a typical input (in our setting, 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 a typical input (in our setting, 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**
Why $k$-CLIQUE may not be FPT?

$k$-CLIQUE: Solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$
Why \textbf{k-Clique} may not be FPT?

\textbf{k-Clique}: Solvable in time \( k^2 \cdot n^k = f(k) \cdot n^{g(k)} \)

Why \textbf{k-Clique} may not be FPT?
Why $k$-CLIQUE may not be FPT?

$k$-CLIQUE: Solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$

Why $k$-CLIQUE may not be FPT?

So far, nobody has managed to find an FPT algorithm for $k$-CLIQUE. (also, nobody has found a poly-time algorithm for 3-SAT)
**Why $k$-CLIQUE may not be FPT?**

$k$-CLIQUE: Solvable in time $k^2 \cdot n^k = f(k) \cdot n^{g(k)}$

So far, nobody has managed to find an FPT algorithm for $k$-CLIQUE. (also, nobody has found a poly-time algorithm for 3-SAT)

Working hypothesis of parameterized complexity: $k$-CLIQUE is not FPT

(in classical complexity: 3-SAT cannot be solved in poly-time)
How to transfer hardness among parameterized problems?

Let \( A, B \) be two parameterized problems.
How to transfer hardness among parameterized problems?

Let $A, B$ be two parameterized problems.

A **parameterized reduction** from $A$ to $B$ is an algorithm such that:

- Instance $(x, k)$ of $A$ has time $f(k) \cdot |x|^{O(1)}$.
- Instance $(x', k')$ of $B$ is a Yes-instance of $A$ if and only if $(x', k')$ is a Yes-instance of $B$.
- $k' \leq g(k)$ for some computable function $g: \mathbb{N} \rightarrow \mathbb{N}$.

$W[1]$-hard problem: $\exists$ parameterized reduction from $k$-Clique to it.


$W[i]$-hard: strong evidence of not being FPT.

Hypothesis: FPT $\neq W[1]$.
How to transfer hardness among parameterized problems?

Let $A, B$ be two parameterized problems.

A parameterized reduction from $A$ to $B$ is an algorithm such that:

1. $(x, k)$ is a $\text{YES}$-instance of $A$ $\iff$ $(x', k')$ is a $\text{YES}$-instance of $B$.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$. 

Instance $(x, k)$ of $A$ \hspace{1cm} time $f(k) \cdot |x|^{O(1)}$ \hspace{1cm} Instance $(x', k')$ of $B$
How to transfer hardness among parameterized problems?

Let $A, B$ be two parameterized problems.

A parameterized reduction from $A$ to $B$ is an algorithm such that:

1. Instance $(x, k)$ of $A$ is a Yes-instance of $A$ $\iff$ $(x', k')$ is a Yes-instance of $B$.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

**W[1]-hard** problem: $\exists$ parameterized reduction from $k$-Clique to it.

**W[2]-hard** problem: $\exists$ param. reduction from $k$-Dominating Set to it.
How to transfer hardness among parameterized problems?

Let $A, B$ be two parameterized problems.

A parameterized reduction from $A$ to $B$ is an algorithm such that:

1. (x, k) is a Yes-instance of $A$ ⇔ (x', k') is a Yes-instance of $B$.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$.

$W[1]$-hard problem: $\exists$ parameterized reduction from $k$-CLIQUE to it.


$W[i]$-hard: strong evidence of not being FPT.
How to transfer hardness among parameterized problems?

Let $A, B$ be two parameterized problems.

A parameterized reduction from $A$ to $B$ is an algorithm such that:

1. Instance $(x, k)$ of $A$ time $f(k) \cdot |x|^{O(1)}$ Instance $(x', k')$ of $B$

1. $(x, k)$ is a $\text{YES}$-instance of $A \iff (x', k')$ is a $\text{YES}$-instance of $B$.
2. $k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

$W[1]$-hard problem: $\exists$ parameterized reduction from $k$-$\text{CLIQUE}$ to it.

$W[2]$-hard problem: $\exists$ param. reduction from $k$-$\text{DOMINATING SET}$ to it.

$W[i]$-hard: strong evidence of not being FPT. Hypothesis: $\text{FPT} \neq W[1]$
Kernelization

**Idea** polynomial-time preprocessing.
A kernel for a parameterized problem $A$ is an algorithm such that:

Instance $(x, k)$ of $A$ \hspace{2cm} \text{polynomial time} \hspace{2cm} \text{Instance } (x', k') \text{ of } A

If $g$ is a polynomial (linear), then we have a polynomial (linear) kernel.

Fact: A problem is FPT $\iff$ it admits a kernel.
Kernelization

Idea polynomial-time preprocessing.

A kernel for a parameterized problem $A$ is an algorithm such that:

1. $(x, k)$ is a $\text{YES}$-instance of $A$ $\iff$ $(x', k')$ is a $\text{YES}$-instance of $A$.
2. $|x'| + k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$. 
A kernel for a parameterized problem $A$ is an algorithm such that:

1. $(x, k)$ is a $\text{Yes}$-instance of $A$ $\iff$ $(x', k')$ is a $\text{Yes}$-instance of $A$.
2. $|x'| + k' \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

The function $g$ is called the size of the kernel.

If $g$ is a polynomial (linear), then we have a polynomial (linear) kernel.
**Kernelization**

**Idea** polynomial-time preprocessing.

A **kernel** for a parameterized problem $A$ is an algorithm such that:

1. $\langle x, k \rangle$ is a $\text{YES}$-instance of $A \iff \langle x', k' \rangle$ is a $\text{YES}$-instance of $A$.
2. $|x'| + k' \leq g(k)$ for some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$.

The function $g$ is called the **size** of the kernel.

If $g$ is a **polynomial** (linear), then we have a **polynomial** (linear) kernel.

**Fact:** A problem is FPT $\iff$ it admits a kernel.
Do all FPT problems admit polynomial kernels?

Fact: A problem is FPT $\iff$ it admits a kernel

Do all FPT problems admit polynomial kernels?
Do all FPT problems admit polynomial kernels?

**Fact:** A problem is FPT \(\iff\) it admits a kernel

Do all FPT problems admit polynomial kernels? **NO!**

**Theorem (Bodlaender, Downey, Fellows, Hermelin. 2009)**

Deciding whether a graph has a Path with \(\geq k\) vertices is FPT but does not admit a polynomial kernel, unless NP \(\subseteq\) coNP/poly.
Typical approach to deal with a parameterized problem

Parameterized problem $L$

$k$-Clique

$k$-Vertex Cover

$k$-Path

Vertex $k$-Coloring
Typical approach to deal with a parameterized problem

Parameterized problem $L$

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

$XP$ $para$-NP-hard
Typical approach to deal with a parameterized problem

Parameterized problem $L$

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

$\text{XP}$

- para-NP-hard

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring
Typical approach to deal with a parameterized problem

Parameterized problem $L$

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

XP

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

W[1]-hard

FPT
Typical approach to deal with a parameterized problem

Parameterized problem $L$

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

XP

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

W[1]-hard

- $k$-Clique

FPT

- $k$-Vertex Cover
- $k$-Path
Typical approach to deal with a parameterized problem

Parameterized problem $L$

- $k$-CLIQUE
- $k$-VERTEX COVER
- $k$-PATH
- VERTEX $k$-COLORING

$\text{XP}$

- $k$-CLIQUE
- $k$-VERTEX COVER
- $k$-PATH

$W[1]$-hard

- $k$-CLIQUE

$\text{FPT}$

- $k$-VERTEX COVER
- $k$-PATH

poly kernel

no poly kernel
Typical approach to deal with a parameterized problem

Parameterized problem $L$

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path
- Vertex $k$-Coloring

XP

- $k$-Clique
- $k$-Vertex Cover
- $k$-Path

para-NP-hard

- Vertex $k$-Coloring

W[1]-hard

- $k$-Clique

FPT

- $k$-Vertex Cover
- $k$-Path

poly kernel

- $k$-Vertex Cover

no poly kernel

- $k$-Path
1 Introduction
   - Parameterized complexity
   - Treewidth

2 FPT algorithms parameterized by treewidth

3 The $\mathcal{F}$-DELETION problem
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:

[Figure by Julien Baste]
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 \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:

[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.

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 \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.

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 \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]
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.
Treewidth via $k$-trees

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.

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

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

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.
Why treewidth?

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 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.
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, ...).
Next section is...

1 Introduction
   • Parameterized complexity
   • Treewidth

2 FPT algorithms parameterized by treewidth

3 The \textsc{\$F\$-Deletion} problem
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 \(k\), ...
Treewidth behaves very well algorithmically

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

Example: \(\text{DomSet}(S) : \left[ \forall v \in V(G) \setminus S, \exists u \in S : \{u, v\} \in E(G) \right] \)
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 $f(\text{tw}) \cdot n$ on graphs on $n$ vertices and treewidth at most $\text{tw}$.

In parameterized complexity: FPT parameterized by treewidth.
Treewidth behaves very well algorithmically.

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

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

**Theorem (Courcelle. 1990)**

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

In parameterized complexity: FPT parameterized by treewidth.

**Examples:** Vertex Cover, Dominating Set, Hamiltonian Cycle, Clique, Independent Set, \( k \)-Coloring for fixed \( k \), ...
Is it enough to prove that a problem is FPT?

Typically, Courcelle’s theorem allows to prove that a problem is FPT...

\[ f(tw) \cdot n^{O(1)} \]
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!

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

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

This is a very active area in parameterized complexity.
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!

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

**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.
Lower bounds on the running times of FPT algorithms

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

\[
\begin{align*}
\text{SETH} & \implies \text{ETH} \implies \text{FPT} \neq \text{W}[1] \implies \text{P} \neq \text{NP}
\end{align*}
\]
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] $\Rightarrow$ 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 $2^{o(\sqrt{k})} \cdot n^{O(1)}$. 
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 \( q \).
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: $2^{tw}$ 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: $2^{tw}$ 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: $2^{tw}$ 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):

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

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.

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):

$2 \Omega (tw \cdot \log tw)$ choices

The "natural" DP algorithms provide only time $2\Omega (tw \cdot \log tw)$.
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):

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

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

24
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): $2 \cdot O\left(\frac{\text{tw} \cdot \log \text{tw}}{n^{O\left(\frac{1}{\text{tw}}\right)}}\right)$ choices.

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

$$2^O(tw \cdot \log tw) $$

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.

- 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...

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 \( 2^{O(tw \cdot \log tw)} \cdot n^{O(1)} \).
Two types of behavior

There seem to be two behaviors for problems parameterized by treewidth:

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

- **Connectivity problems:**
  \[ 2^{O(tw \cdot \log tw)} \cdot n^{O(1)} \]
  
  \textbf{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.

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

Representative sets in matroids: [Fomin, Lokshtanov, Saurabh. 2014]
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.

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: [Cygăn, 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).

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

Representative sets in matroids: [Fomin, Lokshtanov, Saurabh. 2014]
End of the story?

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

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...
1 Introduction
   - Parameterized complexity
   - Treewidth

2 FPT algorithms parameterized by treewidth

3 The $\mathcal{F}$-DELETION problem
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.

*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 deg $\leq 2$.

Therefore: \[ H \text{ topological minor of } G \implies H \text{ minor of } G \]
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 degree $\leq 2$.

- Therefore: $H$ topological minor of $G \not\equiv H$ minor of $G$
The $\mathcal{F}$-M-DELETION problem

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

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

\[ \mathcal{F} = \{K_2\}: \text{Vertex Cover.} \]
The $\mathcal{F}$-M-Deletion problem

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

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

**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 $2^{\Theta(\text{tw})} \cdot n^{O(1)}$.

- $\mathcal{F} = \{K_5, K_3, C_4\}$: Vertex Planarization.
  
  Solvable in time $2^{\Theta(\text{tw} \cdot \log \text{tw})} \cdot n^{O(1)}$.

\[\text{[Cut&Count. 2011]}\]

\[\text{[Jansen, Lokshtanov, Saurabh. 2014 + Pilipczuk. 2015]}\]
The $\mathcal{F}$-$\text{M-Deletion}$ problem

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

<table>
<thead>
<tr>
<th>$\mathcal{F}$-$\text{M-Deletion}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A graph $G$ and an integer $k$.</td>
</tr>
<tr>
<td><strong>Parameter:</strong> The treewidth $\text{tw}$ of $G$.</td>
</tr>
<tr>
<td><strong>Question:</strong> Does $G$ contain a set $S \subseteq V(G)$ with $</td>
</tr>
</tbody>
</table>

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

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

Let $F$ be a fixed finite collection of graphs.

<table>
<thead>
<tr>
<th>$F$-M-Deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A graph $G$ and an integer $k$.</td>
</tr>
<tr>
<td><strong>Parameter:</strong> The treewidth $tw$ of $G$.</td>
</tr>
<tr>
<td><strong>Question:</strong> Does $G$ contain a set $S \subseteq V(G)$ with $</td>
</tr>
</tbody>
</table>

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

- $F = \{K_3\}$: 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 $tw$ 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. Solvable in time $2^{\Theta(tw \cdot \log tw)} \cdot n^{O(1)}$. [Jansen, Lokshpanov, 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 $tw$ 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?
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?
Covering topological minors

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

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

\textbf{Input:} A graph $G$ and an integer $k$.
\textbf{Parameter:} The treewidth $\text{tw}$ 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 graph in $\mathcal{F}$ as a \textbf{minor}?

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

\textbf{Input:} A graph $G$ and an integer $k$.
\textbf{Parameter:} The treewidth $\text{tw}$ 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 graph in $\mathcal{F}$ as a \textbf{topol. minor}?

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

\textbf{FPT} by Courcelle’s Theorem.
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.
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 $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$.

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

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

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

$F$-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...

---

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

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

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

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

$^1$Planar collection $\mathcal{F}$: contains at least one planar graph.
Summary of our results: arXiv 1704.07284+1907.04442

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

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

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

---

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

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

$G$ planar: $\mathcal{F}$-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.)

---

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

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

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

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

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

---

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

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

$G$ planar: $\mathcal{F}$-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.)

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

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

---

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

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

\( \mathcal{G} \) planar: \( \mathcal{F} \)-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.)

\( \mathcal{F} \): \( \mathcal{F} \)-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...

---

\(^1\)Planar collection \( \mathcal{F} \): contains at least one planar graph.
A dichotomy for hitting a connected minor

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 $H \preceq c$.

In both cases, the running time is asymptotically optimal under the ETH.
A dichotomy for hitting a connected minor

Let $H$ be a connected graph.

**Theorem**

Let $H$ be a connected graph.
A dichotomy for hitting a connected minor

Theorem

Let $H$ be a connected graph.
The $\{H\}$-$\text{M-Deletion}$ problem is solvable in time

$$2^{O(tw) \cdot n^{O(1)}}$$

if $H \preceq_c \begin{array}{c} \includegraphics{line_graph} \end{array}$ or $H \preceq_c \begin{array}{c} \includegraphics{square_graph} \end{array}$. 

In both cases, the running time is asymptotically optimal under the ETH.
A dichotomy for hitting a connected minor

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

In both cases, the running time is asymptotically optimal under the ETH.
Theorem

Let $H$ be a connected graph.

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

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

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

In both cases, the running time is asymptotically optimal under the ETH.
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 \( 2\Theta(tw) \) (resp. \( 2\Theta(tw \cdot \log tw) \)). For \( \{ H \} \)-TM-Deletion, \( K_{1,4} \) should be on the left.
A compact statement for a single connected graph

All these cases can be succinctly described as follows:
All these cases can be succinctly described as follows:

- All graphs on the **left** are *contractions* of \( P_2 \) or \( P_3 \).
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 \(P_2\) or \(P_3\).
- All graphs on the **right** are not contractions of \(P_2\) or \(P_3\).
We have three types of results

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

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:
     - [Bodlaender, Cygan, Kratsch, Nederlof. 2013]
     - [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 $2^{O((\text{tw} \cdot \log \text{tw}))} \cdot n^{O(1)}$.
   - $\mathcal{F}$ planar: time $2^{O((\text{tw} \cdot \log \text{tw}))} \cdot n^{O(1)}$.
   - $\mathcal{F} \text{ planar}$: time $2^{O((\text{tw} \cdot \log \text{tw}))} \cdot n^{O(1)}$.
   - $G$ planar: time $2^{O(\text{tw})} \cdot n^{O(1)}$.

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

We build on the machinery of bounded graphs and representatives:

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

Extra: Bidimensionality, irrelevant vertices, protrusion decompositions...
Some ideas of the general algorithms

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

- Every $\mathcal{F}$: 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 $O(\mathcal{F}(t))$. 

$\text{G}'$

$GB$

$B$
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 $C$:

$$p(G, C) = \min\{|S| : S \subseteq V(G) \land \text{folio}(G - S) = C\}$$
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 $\mathcal{O}_\mathcal{F}(t)$.

- 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 $2^{\mathcal{O}_\mathcal{F}(t \cdot \log t)}$. 
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 $\mathcal{O}(t)$.

- 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 $G$,

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

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

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


Algorithm for a planar collection $\mathcal{F}$

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

$$G_1 \equiv (\mathcal{F}, t) G_2 \iff \forall G' \in B_t, \mathcal{F} \preceq m G' \oplus G_1 \iff \mathcal{F} \preceq m G' \oplus G_2.$$ 

$R(\mathcal{F}, t)$: 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 $|R(\mathcal{F}, t)| = 2^{O(\mathcal{F}(t) \cdot \log t)}$.

Planarity!

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

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

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

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

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

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

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

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

Planarity! The number of labeled graphs of size $\leq t$ and $tw \leq h$ is $2^O(h(t) \cdot \log t)$.

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

$\text{skip}$
Algorithm for a planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv(\mathcal{F},t)$ on $t$-boundaried graphs:
  - $G_1 \equiv(\mathcal{F},t) G_2$ 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 $\equiv(\mathcal{F},t)$.

Planarity! 

\[ \# \text{labeled graphs of size } \leq t \text{ and } tw \leq h \text{ is } 2^{O(h \cdot (t \cdot \log t))}. \]

\[ Baste, Noy, S. 2017 \]

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

\[ 40 \]
Algorithm for a planar collection $\mathcal{F}$

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv(\mathcal{F},t)$ on $t$-boundarybrid 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 representative 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 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 $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$. 

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

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

#40
Algorithm for a planar collection $\mathcal{F}$

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

  $G_1 \equiv(\mathcal{F}, t) G_2$ 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 $|\mathcal{R}(\mathcal{F}, t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.

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

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

- For a fixed $\mathcal{F}$, we define an equivalence relation $\equiv(\mathcal{F},t)$ on $t$-boundaried graphs:
  
  $G_1 \equiv(\mathcal{F},t) G_2$ 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 $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.  
  
  # labeled graphs of size $\leq t$ and $\text{tw} \leq h$ is $2^{O_h(t \cdot \log t)}$.  
  
  [Baste, Noy, S. 2017]

- Planarity!

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

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

- $G_1 \equiv(\mathcal{F},t) G_2$ 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 $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.  

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

  [Baste, Noy, S. 2017]

- Planarity!

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

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

  $G_1 \equiv(\mathcal{F},t) G_2$ 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 $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.  

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

  [Baste, Noy, S. 2017]

- Planarity!

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

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

  $G_1 \equiv(\mathcal{F},t) G_2$ 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 $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$.  

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

  [Baste, Noy, S. 2017]

- Planarity!

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

Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F}, t)$, $|V(R)| = O(\mathcal{F}(t))$. We are done: $|\mathcal{R}(\mathcal{F}, t)| = 2^{O(\mathcal{F}(t) \cdot \log t)}$ and the same DP works!

Flat Wall Theorem:
As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_{\mathcal{F}}$, $R \setminus B$ contains a large flat wall, where we can find an irrelevant vertex. $R$ has a treewidth modulator of size $O(t)$ containing its boundary $B$. We can then find a linear protrusion decomposition of $R$.

By applying protrusion reduction, we obtain that $|V(R)| = O(\mathcal{F}(t))$. 

41
Algorithm for any collection $\mathcal{F}$

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

- $\mathcal{R}(\mathcal{F},t)$: set of minimum-size representatives of $\equiv(\mathcal{F},t)$.
- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F},t)$,
  $$|V(R)| = \mathcal{O}_\mathcal{F}(t).$$
Algorithm for any collection $\mathcal{F}$

- $\mathcal{R}(\mathcal{F},t)$: set of minimum-size representatives of $\equiv(\mathcal{F},t)$.
- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F},t)$,
  \[ |V(R)| = O_{\mathcal{F}}(t). \]
- We are done: $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$ and the same DP works!
Algorithm for any collection $\mathcal{F}$

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

- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F},t)$,
  $$|V(R)| = O_{\mathcal{F}}(t).$$

- We are done: $|\mathcal{R}(\mathcal{F},t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$ and the same DP works!

- Flat Wall Theorem:
Algorithm for any collection $\mathcal{F}$

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

- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F}, t)$,
  \[ |V(R)| = \mathcal{O}_\mathcal{F}(t). \]

- We are done: $|\mathcal{R}(\mathcal{F}, t)| = 2^{\mathcal{O}_\mathcal{F}(t \cdot \log t)}$ and the same DP works!

- Flat Wall Theorem: As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_\mathcal{F}$,
Algorithm for any collection $\mathcal{F}$

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

Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F}, t)$,

$$|V(R)| = O_{\mathcal{F}}(t).$$

We are done: $|\mathcal{R}(\mathcal{F}, t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$ and the same DP works!

- Flat Wall Theorem: As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_{\mathcal{F}}$, $R \setminus B$ contains a large flat wall,
Algorithm for any collection $\mathcal{F}$

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

- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F},t)$,
  \[
  |V(R)| = \mathcal{O}_\mathcal{F}(t).
  \]

- We are done: $|\mathcal{R}(\mathcal{F},t)| = 2^{\mathcal{O}_\mathcal{F}(t \cdot \log t)}$ and the same DP works!

- **Flat Wall Theorem**: As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_\mathcal{F}$, $R \setminus B$ contains a large flat wall, where we can find an irrelevant vertex.
Algorithm for any collection $\mathcal{F}$

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

- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F}, t)$,
  $$|V(R)| = O_{\mathcal{F}}(t).$$

- We are done: $|\mathcal{R}(\mathcal{F}, t)| = 2^{O_{\mathcal{F}}(t \cdot \log t)}$ and the same DP works!

- Flat Wall Theorem: As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_{\mathcal{F}}$, $R \setminus B$ contains a large flat wall, where we can find an irrelevant vertex.

- $R$ has a treewidth modulator of size $O(t)$ containing its boundary $B$. 
Algorithm for any collection $\mathcal{F}$

- $\mathcal{R}^{(\mathcal{F}, t)}$: set of minimum-size representatives of $\equiv(\mathcal{F}, t)$.
- Suppose that we can prove that, for every $R \in \mathcal{R}^{(\mathcal{F}, t)}$,
  $$|V(R)| = \mathcal{O}_\mathcal{F}(t).$$
- We are done: $|\mathcal{R}^{(\mathcal{F}, t)}| = 2^{\mathcal{O}_\mathcal{F}(t \cdot \log t)}$ and the same DP works!

- Flat Wall Theorem: As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_{\mathcal{F}}$, $R \setminus B$ contains a large flat wall, where we can find an irrelevant vertex.
- $R$ has a treewidth modulator of size $\mathcal{O}(t)$ containing its boundary $B$.
- We can then find a linear protrusion decomposition of $R$. 
Algorithm for any collection $\mathcal{F}$

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

- Suppose that we can prove that, for every $R \in \mathcal{R}(\mathcal{F}, t)$,
  \[ |V(R)| = \mathcal{O}_\mathcal{F}(t). \]

- We are done: $|\mathcal{R}(\mathcal{F}, t)| = 2^{\mathcal{O}_\mathcal{F}(t \cdot \log t)}$ and the same DP works!

- Flat Wall Theorem: As $R$ is $\mathcal{F}$-minor-free, if $\text{tw}(R \setminus B) > c_{\mathcal{F}}$, $R \setminus B$ contains a large flat wall, where we can find an irrelevant vertex.

- $R$ has a treewidth modulator of size $\mathcal{O}(t)$ containing its boundary $B$.

- We can then find a linear protrusion decomposition of $R$.

- By applying protrusion reduction, we obtain that $|V(R)| = \mathcal{O}_\mathcal{F}(t)$. 
Algorithm for any collection $\mathcal{F}$
Algorithm for any collection $\mathcal{F}$
Algorithm for any 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]

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

$h = f(J)$

$R \in \mathcal{R}_h^{(t)}$

$R$ contains no irrelevant vertex [Theorem 19]

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

$P_{h,r}$ is bidimensional [Lemma 8]

$P_{h,r}$ is separable [Lemma 9]

$R$ has a treewidth modulator of size $O(t)$ containing the boundary [Lemma 22]

[34]

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

Reduce protrusions [5]

Linear protrusion decomposition of $R$ [Lemma 24]

Sparsity of the representatives

$|\mathcal{R}_h^{(t)}| = 2^{O_h(t \log t)}$ [Corollary 27]

DP algorithm from [5]

Algorithm in time $O^*(2^{O_h(\text{tw} \cdot \log \text{tw})})$ 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
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 $2^{O(F(tw))} \cdot n^{O(1)}$.

We can extend this algorithm to input graphs $G$ embedded in arbitrary surfaces by using surface-cut decompositions.
Algorithm when the input graph $G$ is planar

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

- 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**.

![Sphere-cut decomposition diagram](image-url)
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**.

![Diagram](image_url)

- 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 $2^{O(\mathcal{F}(tw))} \cdot n^{O(1)}$. 

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]
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 $2^{\Theta(tw \cdot \log tw)}$?

Lower bounds for 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.

Conjecture

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

45
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 $2 \Theta(tw \cdot \log tw)$?

Lower bounds for 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.

Conjecture

For every family $\mathcal{F}$, the $\mathcal{F}$-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:

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

  - Lower bounds for 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.

- Conjecture For every family $\mathcal{F}$, the $\mathcal{F}$-**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).

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.

Conjecture
- For every family $\mathcal{F}$, the $\mathcal{F}$-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): $2^{\Theta(tw)}$ or $2^{\Theta(tw \cdot \log tw)}$?

- Lower bounds for 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.

Conjecture
- For every family $\mathcal{F}$, the $\mathcal{F}$-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): $2^{\Theta(tw)}$ or $2^{\Theta(tw \cdot \log tw)}$?
  - Lower bounds for families $\mathcal{F}$ containing disconnected graphs.
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 $\mathcal{F}$.

Considering the **minor** version:

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

- Lower bounds for 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 $\mathcal{F}$-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 $2^{\Theta(tw \cdot \log tw)}$?

  - Lower bounds for 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 $2^{\Theta(tw \cdot \log tw)}$?
  - Lower bounds for 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 $2^{\Theta(tw \cdot \log tw)}$?
  - Lower bounds for 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 $\mathcal{F}$-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 $2^{\Theta(tw \cdot \log tw)}$?
  - Lower bounds for 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.
  - **Conjecture** For every family $\mathcal{F}$, the $\mathcal{F}$-TM-DELETION problem is solvable in time $2^{O(tw \cdot \log tw)} \cdot n^{O(1)}$. 
For topological minors, there is (at least) one change

\[ 2^{Θ(tw)} \]

\[ 2^{Θ(tw \cdot \log tw)} \]

\[ P_5 \]

\[ \text{diamond} \]
\[ \text{K}_4 \]
\[ \text{C}_5 \]
\[ \text{K}_{1,4} \]

\[ \text{K}_3 \cup 2\text{K}_1 \]
\[ \text{K}_5-e \]
\[ \\text{W}_4 \]
\[ \text{P}_3 \cup 2\text{K}_1 \]

\[ \text{P}_2 \cup \text{P}_3 \]
\[ \text{gem} \]
\[ \text{house} \]
\[ \text{K}_5 \]

\[ \text{px} \]
\[ \text{kite} \]
\[ \text{dart} \]
\[ \text{K}_{2,3} \]

\[ \text{P}_2 \]
\[ \text{P}_3 \]
\[ \text{P}_4 \]
\[ \text{P}_5 \]
\[ \text{C}_3 \]
\[ \text{C}_4 \]
\[ \text{claw} \]
\[ \text{paw} \]
\[ \text{chair} \]
\[ \text{banner} \]
\[ \text{bull} \]
\[ \text{butterfly} \]
\[ \text{cricket} \]
\[ \text{co-banner} \]
Gràcies!