Single-exponential algorithms and linear kernels via protrusion decompositions

Eun Jung Kim$^1$  Christophe Paul$^2$  Ignasi Sau$^2$
Alexander Langer$^3$  Felix Reidl$^3$  Peter Rossmanith$^3$  Somnath Sikdar$^3$

arXiv/1207.0835, 2013

$^1$ CNRS, LAMSADE, Paris (France)
$^2$ CNRS, LIRMM, Montpellier (France)
$^3$ Department of Computer Science, RWTH Aachen University (Germany)
Outline of the talk

1 Preliminaries

2 Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   - Motivation and our result
   - Sketch of proof
   - Further research

4 Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
1 Preliminaries

2 Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3 Single-exponential algorithm for Planar-\( \mathcal{F} \)-Deletion
   - Motivation and our result
   - Sketch of proof
   - Further research

4 Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
Some words on parameterized complexity

- **Idea**: given an NP-hard problem with input size $n$, fix one parameter $k$ of the input to see whether the problem gets more “tractable”.

  **Example**: the size of a **Vertex Cover**.
Some words on parameterized complexity

- **Idea**: given an NP-hard problem with input size $n$, fix one parameter $k$ of the input to see whether the problem gets more “tractable”.

  **Example**: the size of a Vertex Cover.

- Given a (NP-hard) problem with input of size $n$ and a parameter $k$, a fixed-parameter tractable (FPT) algorithm runs in time

$$f(k) \cdot n^{O(1)},$$

for some function $f$.

**Examples**: $k$-Vertex Cover, $k$-Longest Path.
Some words on parameterized complexity

- **Idea**: given an NP-hard problem with input size $n$, fix one parameter $k$ of the input to see whether the problem gets more “tractable”.

  **Example**: the size of a **Vertex Cover**.

- Given a (NP-hard) problem with input of size $n$ and a parameter $k$, a fixed-parameter tractable (FPT) algorithm runs in time

  $$f(k) \cdot n^{O(1)}, \text{ for some function } f.$$ 

  **Examples**: $k$-**Vertex Cover**, $k$-**Longest Path**.

- A single-exponential parameterized algorithm is an FPT algo s.t.

  $$f(k) = 2^{O(k)}.$$
Many hard algorithmic graph problems become easier if one is able to find a suitable decomposition of the input graph.
The decomposition paradigm — “Divide et impera”

Many **hard algorithmic graph problems** become **easier** if one is able to find a **suitable decomposition** of the input graph.

Some famous examples:

- **PTAS and exact subexponential algorithms** based on finding separators of size $O(\sqrt{n})$ on planar graphs.  
  
  [Baker’s approach]
The decomposition paradigm — “Divide et impera”

Many hard algorithmic graph problems become easier if one is able to find a suitable decomposition of the input graph.

Some famous examples:

- **PTAS and exact subexponential algorithms** based on finding separators of size $O(\sqrt{n})$ on planar graphs. [Baker’s approach]

- **Linear-time algorithms** for problems expressible in MSOL on graphs of bounded treewidth. [Coucelle’s theorem]
Many **hard algorithmic graph problems** become **easier** if one is able to find a **suitable decomposition** of the input graph.

Some famous examples:

- **PTAS and exact subexponential algorithms** based on finding separators of size $O(\sqrt{n})$ on planar graphs.  
  
  [Baker’s approach]

- **Linear-time algorithms** for problems expressible in MSOL on graphs of bounded treewidth.  
  
  [Coucelle’s theorem]

- **FPT algorithms** based on the **structural decomposition result of $H$-minor-free graphs**.  
  
  [Graph Minors theory by Robertson and Seymour]
Many hard algorithmic graph problems become easier if one is able to find a suitable decomposition of the input graph.

Some famous examples:

- **PTAS and exact subexponential algorithms** based on finding separators of size $O(\sqrt{n})$ on planar graphs. [Baker’s approach]

- **Linear-time algorithms** for problems expressible in MSOL on graphs of bounded treewidth. [Coucelle’s theorem]

- **FPT algorithms** based on the structural decomposition result of $H$-minor-free graphs. [Graph Minors theory by Robertson and Seymour]

- **Linear-time algorithms** based on modular decompositions.
1 Preliminaries

2 Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   - Motivation and our result
   - Sketch of proof
   - Further research

4 Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
1. Preliminaries

2. Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3. Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   - Motivation and our result
   - Sketch of proof
   - Further research

4. Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
Given a graph $G$, a set $W \subseteq V(G)$ is a $t$-protrusion of $G$ if

$$|\partial_G(W)| \leq t \quad \text{and} \quad tw(G[W]) \leq t$$
Given a graph $G$, a set $W \subseteq V(G)$ is a $t$-protrusion of $G$ if

$$|\partial_G(W)| \leq t \text{ and } tw(G[W]) \leq t$$

The vertex set $W' = W \setminus \partial_G(W)$ is the restricted protrusion of $W$.

We call $\partial_G(W)$ the boundary and $|W|$ the size of $W$. 
Protrusion decompositions

An \((\alpha, t)\)-protrusion decomposition of a graph \(G\) is a partition \(\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell\) of \(V(G)\) such that:

- for every \(1 \leq i \leq \ell\), \(N(Y_i) \subseteq Y_0\);
- for every \(1 \leq i \leq \ell\), \(Y_i \cup N_{Y_0}(Y_i)\) is a \(t\)-protrusion of \(G\);
- \(\max\{\ell, |Y_0|\} \leq \alpha\).

The set \(Y_0\) is called the separating part of \(\mathcal{P}\).
1 Preliminaries

2 Protrusion decompositions
   • Definitions
   • A simple algorithm to compute them

3 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   • Motivation and our result
   • Sketch of proof
   • Further research

4 Linear kernels on graphs without topological minors
   • Motivation and our result
   • Idea of proof
   • Further research
Main (informal) ideas of our algorithm

- Protrusion decompositions have already been used in the literature.

[Bodlaender, Fomin, Lokshtanov, Saurabh, Thilikos '09-12]
Main (informal) ideas of our algorithm

Here we present a new algorithm to compute protrusion decompositions for graphs $G$ that come equipped with a set

$$X \subseteq V(G) \text{ s.t. } tw(G - X) \leq t$$

for some constant $t > 0$.

The set $X$ is called a $t$-treewidth-modulator.
Main (informal) ideas of our algorithm

- Our algorithm marks the bags of a tree-decomposition of $G$. 

- Bloom components

- Bud components

- The set $V(M)$ of vertices contained in marked bags together with $X$ will form the separating part $Y_0$ of the protrusion decomposition.

- Some marked bags will be mapped bijectively into pairwise vertex-disjoint connected subgraphs of $G - X$, each of which has $\geq r$ neighbors in $X$.

- Finally, to guarantee that the connected components of $G - (X \cup V(M))$ form protrusions with small boundary, the set $M$ is closed under taking LCA.
Main (informal) ideas of our algorithm

- Our algorithm marks the bags of a tree-decomposition of $G$.
- Let $r$ be an integer that is also given to the algorithm.
Main (informal) ideas of our algorithm

- Our algorithm marks the bags of a tree-decomposition of $G$.
- Let $r$ be an integer that is also given to the algorithm.
- Given tree-decompositions of the conn. comp. of $G - X$ with $\geq r$ neighbors in $X$, we identify a set of bags $M$ in a bottom-up manner.

Bloom components

Bud components
Main (informal) ideas of our algorithm

- Our algorithm marks the bags of a tree-decomposition of $G$.
- Let $r$ be an integer that is also given to the algorithm.
- Given tree-decompositions of the conn. comp. of $G - X$ with $\geq r$ neighbors in $X$, we identify a set of bags $\mathcal{M}$ in a bottom-up manner.

![Diagram of a tree-decomposition with marked bags and a set $V(M)$ of vertices]

- The set $V(\mathcal{M})$ of vertices contained in marked bags together with $X$ will form the separating part $Y_0$ of the protrusion decomposition.
Main (informal) ideas of our algorithm

- Our algorithm marks the bags of a tree-decomposition of $G$.
- Let $r$ be an integer that is also given to the algorithm.
- Given tree-decompositions of the conn. comp. of $G - X$ with $\geq r$ neighbors in $X$, we identify a set of bags $\mathcal{M}$ in a bottom-up manner.

The set $V(\mathcal{M})$ of vertices contained in marked bags together with $X$ will form the separating part $Y_0$ of the protrusion decomposition.

Some marked bags will be mapped bijectively into pairwise vertex-disjoint connected subgraphs of $G - X$, each of which has $\geq r$ neighbors in $X$. 
Main (informal) ideas of our algorithm

- Our algorithm marks the bags of a tree-decomposition of $G$.
- Let $r$ be an integer that is also given to the algorithm.
- Given tree-decompositions of the conn. comp. of $G - X$ with $\geq r$ neighbors in $X$, we identify a set of bags $\mathcal{M}$ in a bottom-up manner.

The set $V(\mathcal{M})$ of vertices contained in marked bags together with $X$ will form the separating part $Y_0$ of the protrusion decomposition.

Some marked bags will be mapped bijectively into pairwise vertex-disjoint connected subgraphs of $G - X$, each of which has $\geq r$ neighbors in $X$.

Finally, to guarantee that the conn. comp. of $G - (X \cup V(\mathcal{M}))$ form protrusions with small boundary, the set $\mathcal{M}$ is closed under taking LCA.
Description of the bag marking algorithm

**Input**  
$G, X \subseteq V(G)$ s.t. $tw(G - X) \leq t$, and an integer $r > 0$. 

Set $M \leftarrow \emptyset$ as the set of marked bags.

Compute an optimal rooted tree-decomposition $T_C = (T_C, B_C)$ of every connected component $C$ of $G - X$ such that $|N_X(C)| \geq r$.

Repeat the following loop for every rooted tree-decomposition $T_C$:

- While $T_C$ contains an unprocessed bag do:
  - Let $B$ be an unprocessed bag at farthest distance from the root of $T_C$.
  - **LCA marking step** if $B$ is the LCA of two marked bags of $M$:
    - $M \leftarrow M \cup \{B\}$ and remove the vertices of $B$ from every bag of $T_C$.
  - **Bloom-subgraph marking step** else if $G_B$ contains a connected component $C_B$ s.t. $|N_X(C_B)| \geq r$:
    - $M \leftarrow M \cup \{B\}$ and remove the vertices of $B$ from every bag of $T_C$.

- $B$ is now processed.

Return $Y_0 = X \cup V(M)$. 


Description of the bag marking algorithm

**Input** \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

* Set \( \mathcal{M} \leftarrow \emptyset \) as the set of marked bags.
Description of the bag marking algorithm

**Input** \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

- Set \( \mathcal{M} \leftarrow \emptyset \) as the set of marked bags.
- Compute an optimal rooted tree-decomposition \( T_C = (T_C, B_C) \) of every connected component \( C \) of \( G - X \) such that \( |N_X(C)| \geq r \).

⋆ Let \( B \) be an unprocessed bag at farthest distance from the root of \( T_C \).

   ⋆ **LCA marking step**
   - If \( B \) is the LCA of two marked bags of \( \mathcal{M} \):
     - \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \)
     - Remove the vertices of \( B \) from every bag of \( T_C \).
   - **Bloom-subgraph marking step**
   - Else if \( G_B \) contains a connected component \( C_B \) s.t. \( |N_X(C_B)| \geq r \):
     - \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \)
     - Remove the vertices of \( B \) from every bag of \( T_C \).

⋆ Bag \( B \) is now processed.

Return \( Y_0 = X \cup V(M) \).
Description of the bag marking algorithm

Input \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

- Set \( M \leftarrow \emptyset \) as the set of marked bags.
- Compute an optimal rooted tree-decomposition \( T_C = (T_C, B_C) \) of every connected component \( C \) of \( G - X \) such that \( |N_X(C)| \geq r \).
- Repeat the following loop for every rooted tree-decomposition \( T_C \):
  - \( \text{while } T_C \text{ contains an unprocessed bag do:} \)
    - Let \( B \) be an unprocess. bag at farthest distance from the root of \( T_C \).
Description of the bag marking algorithm

**Input** \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

- Set \( \mathcal{M} \leftarrow \emptyset \) as the set of marked bags.
- Compute an optimal rooted tree-decomposition \( \mathcal{T}_C = (T_C, B_C) \) of every connected component \( C \) of \( G - X \) such that \( |N_X(C)| \geq r \).
- Repeat the following loop for every rooted tree-decomposition \( \mathcal{T}_C \):
  - **while** \( \mathcal{T}_C \) contains an unprocessed bag **do**:
    - Let \( B \) be an unprocessed bag at farthest distance from the root of \( \mathcal{T}_C \).
    - **LCA marking step**
      - if \( B \) is the LCA of two marked bags of \( \mathcal{M} \):
        \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
Description of the bag marking algorithm

**Input** \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

- Set \( \mathcal{M} \leftarrow \emptyset \) as the set of marked bags.
- Compute an optimal rooted tree-decomposition \( \mathcal{T}_C = (T_C, B_C) \) of every connected component \( C \) of \( G - X \) such that \( |N_X(C)| \geq r \).
- Repeat the following loop for every rooted tree-decomposition \( \mathcal{T}_C \):
  - **while** \( \mathcal{T}_C \) contains an unprocessed bag **do**:
    - Let \( B \) be an unprocessed bag at farthest distance from the root of \( \mathcal{T}_C \).
    - **LCA marking step**
      - if \( B \) is the LCA of two marked bags of \( \mathcal{M} \):
        - \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
    - **Bloom-subgraph marking step**
      - else if \( G_B \) contains a connected component \( C_B \) s.t. \( |N_X(C_B)| \geq r \):
        - \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
Description of the bag marking algorithm

**Input** \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

- Set \( \mathcal{M} \leftarrow \emptyset \) as the set of marked bags.
- Compute an **optimal** rooted tree-decomposition \( \mathcal{T}_C = (T_C, B_C) \) of every connected component \( C \) of \( G - X \) such that \( |N_X(C)| \geq r \).
- Repeat the following **loop** for every rooted tree-decomposition \( \mathcal{T}_C \):
  - **while** \( \mathcal{T}_C \) contains an **unprocessed bag** **do**:
    - Let \( B \) be an **unprocessed** bag at farthest distance from the root of \( \mathcal{T}_C \).
    - **LCA marking step**
      - if \( B \) is the LCA of two marked bags of \( \mathcal{M} \):
        \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
    - **Bloom-subgraph marking step**
      - else if \( G_B \) contains a connected component \( C_B \) s.t. \( |N_X(C_B)| \geq r \):
        \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
    - Bag \( B \) is now processed.
Description of the bag marking algorithm

**Input** \( G, X \subseteq V(G) \) s.t. \( \text{tw}(G - X) \leq t \), and an integer \( r > 0 \).

- Set \( \mathcal{M} \leftarrow \emptyset \) as the set of marked bags.
- Compute an **optimal** rooted tree-decomposition \( \mathcal{T}_C = (T_C, B_C) \) of every connected component \( C \) of \( G - X \) such that \( |N_X(C)| \geq r \).
- Repeat the following **loop** for every rooted tree-decomposition \( \mathcal{T}_C \):
  - **while** \( \mathcal{T}_C \) contains an **unprocessed bag** **do**:
    - Let \( B \) be an **unprocessed** bag at farthest distance from the root of \( \mathcal{T}_C \).
    - **LCA marking step**
      - if \( B \) is the LCA of two marked bags of \( \mathcal{M} \):
        - \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
    - **Bloom-subgraph marking step**
      - else if \( G_B \) contains a connected component \( C_B \) s.t. \( |N_X(C_B)| \geq r \):
        - \( \mathcal{M} \leftarrow \mathcal{M} \cup \{B\} \) and remove the vertices of \( B \) from every bag of \( \mathcal{T}_C \).
  - **Bag** \( B \) **is now processed**.

**Return** \( Y_0 = X \cup V(\mathcal{M}) \).
Some properties of the bag marking algorithm

**Lemma**

The *bag marking algorithm* can be implemented to run in $O(n)$ time, where the hidden constant depends only on $t$ and $r$. 

*Figure by Felix Reidl*
Some properties of the bag marking algorithm

Given a graph $G$ and a subset $S \subseteq V(G)$, a cluster of $G - S$ is a maximal collection of connected components of $G - S$ with the same neighborhood in $S$. 
Some properties of the bag marking algorithm

Given a graph $G$ and a subset $S \subseteq V(G)$, a cluster of $G - S$ is a maximal collection of connected components of $G - S$ with the same neighborhood in $S$.

**Proposition**

- Let $r, t$ be two positive integers,
- let $G$ be a graph and $X \subseteq V(G)$ such that $\text{tw}(G - X) \leq t$,
- let $Y_0 \subseteq V(G)$ be the output of the algorithm with input $(G, X, r)$, and
- let $Y_1, \ldots, Y_\ell$ be the set of clusters of $G - Y_0$. 

Then $P := Y_0 \cup Y_1 \cup \cdots \cup Y_\ell$ is a $(\max \{\ell, |Y_0|\}, 2t + r)$-protrusion decomp. of $G$. 

(Figure by Felix Reidl)
Some properties of the bag marking algorithm

Given a graph $G$ and a subset $S \subseteq V(G)$, a cluster of $G - S$ is a maximal collection of connected components of $G - S$ with the same neighborhood in $S$.

**Proposition**

- Let $r, t$ be two positive integers,
- let $G$ be a graph and $X \subseteq V(G)$ such that $\text{tw}(G - X) \leq t$,
- let $Y_0 \subseteq V(G)$ be the output of the algorithm with input $(G, X, r)$, and
- let $Y_1, \ldots, Y_\ell$ be the set of clusters of $G - Y_0$.

Then $\mathcal{P} := Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ is a $(\max\{\ell, |Y_0|\}, 2t + r)$-protrusion decomp. of $G$.

(Figure by Felix Reidl)
1 Preliminaries

2 Protrusion decompositions
   • Definitions
   • A simple algorithm to compute them

3 Single-exponential algorithm for **Planar-\mathcal{F}-Deletion**
   • Motivation and our result
   • Sketch of proof
   • Further research

4 Linear kernels on graphs without topological minors
   • Motivation and our result
   • Idea of proof
   • Further research
1 Preliminaries

2 Protrusion decompositions
   • Definitions
   • A simple algorithm to compute them

3 Single-exponential algorithm for **Planar-\(\mathcal{F}\)-Deletion**
   • Motivation and our result
   • Sketch of proof
   • Further research

4 Linear kernels on graphs without topological minors
   • Motivation and our result
   • Idea of proof
   • Further research
The (parameterized) **PLANAR-$\mathcal{F}$-DELETION** problem

Let $\mathcal{F}$ be a finite family of graphs containing at least one planar graph.
The (parameterized) $\text{Planar-} \mathcal{F}\text{-Deletion}$ problem

Let $\mathcal{F}$ be a finite family of graphs containing at least one planar graph.

**Planar-\(\mathcal{F}\)-Deletion**

**Input:** A graph $G$ and a non-negative integer $k$.

**Parameter:** The integer $k$.

**Question:** Does $G$ have a set $X \subseteq V(G)$ such that $|X| \leq k$ and $G - X$ is $H$-minor-free for every $H \in \mathcal{F}$?
The (parameterized) \textsc{Planar-\mathcal{F}-Deletion} problem

Let \mathcal{F} be a finite family of graphs containing \textit{at least one planar graph}.

\textbf{Planar-\mathcal{F}-Deletion}

\textbf{Input:} A graph \(G\) and a non-negative integer \(k\).
\textbf{Parameter:} The integer \(k\).
\textbf{Question:} Does \(G\) have a set \(X \subseteq V(G)\) such that \(|X| \leq k\) and \(G - X\) is \(H\)-minor-free for every \(H \in \mathcal{F}\)?

Some particular cases:

1. \(\mathcal{F} = \{K_2\}\):
   \(\equiv\) \textsc{Vertex Cover}
   \(\equiv\) \textsc{Treewidth-zero Vertex Deletion}

2. \(\mathcal{F} = \{K_3\}\):
   \(\equiv\) \textsc{Feedback Vertex Set}
   \(\equiv\) \textsc{Treewidth-one Vertex Deletion}

3. \(\mathcal{F} = \{K_4\}\):
   \(\equiv\) \textsc{Treewidth-two Vertex Deletion}
How fast can $\text{PLANAR-}F\text{-DELETION}$ be solved?

Particular cases:

1. $F = \{K_2\}$
   
   $O^*(1.2738k)$
   
   [Chen, Fernau, Kanj, Xia '10]

2. $F = \{K_3\}$
   
   $O^*(3.83k)$
   
   [Cao, Chen, Liu '10]

3. $F = \{\theta_r\}$
   
   $O^*(ck)$
   
   [Joret, Paul, S., Saurabh, Thomassé '11]

4. $F = \{K_4\}$
   
   $O^*(ck)$
   
   [Kim, Paul, Philip '12]

General case:

$\text{PLANAR-}F\text{-DELETION}$ is $FPT$.

[Robertson and Seymour's Graph Minors theory]

2

$O(k \log k) \cdot n^O(1)$-time algorithm based on standard DP.

[Fomin, Lokshtanov, Misra, Saurabh '11]

2

$O(k \log k)$-time algorithm.

[Fomin, Lokshtanov, Misra, Saurabh '12]

$O(k \log 2 n)$-time algorithm for $\text{Planar-Connected-}F\text{-Deletion}$.

[Fomin, Lokshtanov, Misra, Saurabh '12]
How fast can Planar-$\mathcal{F}$-Deletion be solved?

Particular cases:

- $\mathcal{F} = \{K_2\}$ \hspace{1cm} $O^*(1.2738^k)$ \hspace{1cm} [Chen, Fernau, Kanj, Xia '10]
- $\mathcal{F} = \{K_3\}$ \hspace{1cm} $O^*(3.83^k)$ \hspace{1cm} [Cao, Chen, Liu '10]
- $\mathcal{F} = \{\theta_r\}$ \hspace{1cm} $O^*(c^k)$ \hspace{1cm} [Joret, Paul, S., Saurabh, Thomassé '11]
- $\mathcal{F} = \{K_4\}$ \hspace{1cm} $O^*(c^k)$ \hspace{1cm} [Kim, Paul, Philip '12]
How fast can **Planar-**$\mathcal{F}$-Deletion** be solved?**

**Particular cases:**

- $\mathcal{F} = \{K_2\}$  \[ O^*(1.2738^k) \]  \[ [\text{Chen, Fernau, Kanj, Xia '10}] \]
- $\mathcal{F} = \{K_3\}$  \[ O^*(3.83^k) \]  \[ [\text{Cao, Chen, Liu '10}] \]
- $\mathcal{F} = \{\theta_r\}$  \[ O^*(c^k) \]  \[ [\text{Joret, Paul, S., Saurabh, Thomassé '11}] \]
- $\mathcal{F} = \{K_4\}$  \[ O^*(c^k) \]  \[ [\text{Kim, Paul, Philip '12}] \]

**General case:**

- **Planar-**$\mathcal{F}$-Deletion** is FPT.**  \[ [\text{Roberston and Seymour's Graph Minors theory}] \]
How fast can \textsc{Planar-\textit{F}-Deletion} be solved?

Particular cases:

- \( \mathcal{F} = \{K_2\} \) \( O^*(1.2738^k) \) \[Chen, Fernau, Kanj, Xia '10\]
- \( \mathcal{F} = \{K_3\} \) \( O^*(3.83^k) \) \[Cao, Chen, Liu '10\]
- \( \mathcal{F} = \{\theta_r\} \) \( O^*(c^k) \) \[Joret, Paul, S., Saurabh, Thomassé '11\]
- \( \mathcal{F} = \{K_4\} \) \( O^*(c^k) \) \[Kim, Paul, Philip '12\]

General case:

- \textsc{Planar-\textit{F}-Deletion} is \textsc{FPT}. \[Roberston and Seymour’s Graph Minors theory\]
- \( 2^{O(k \log k)} \cdot n^{O(1)} \) -time algorithm based on standard DP.
How fast can \textsc{Planar-$F$-Deletion} be solved?

Particular cases:

- $F = \{K_2\}$ \quad $O^*(1.2738^k)$ \quad [Chen, Fernau, Kanj, Xia '10]
- $F = \{K_3\}$ \quad $O^*(3.83^k)$ \quad [Cao, Chen, Liu '10]
- $F = \{\theta_r\}$ \quad $O^*(c^k)$ \quad [Joret, Paul, S., Saurabh, Thomassé '11]
- $F = \{K_4\}$ \quad $O^*(c^k)$ \quad [Kim, Paul, Philip '12]

General case:

- \textsc{Planar-$F$-Deletion} is \textbf{FPT}. \quad [Roberston and Seymour's Graph Minors theory]
- $2^{O(k \log k)} \cdot n^{O(1)}$ -time algorithm based on standard DP.
- $2^{O(k \log k)} \cdot n^2$ -time algorithm. \quad [Fomin, Lokshtanov, Misra, Saurabh '11]
How fast can Planar-\(\mathcal{F}\)-Deletion be solved?

**Particular cases:**

- \(\mathcal{F} = \{K_2\}\) \(O^*(1.2738^k)\)  
  [Chen, Fernau, Kanj, Xia ’10]
- \(\mathcal{F} = \{K_3\}\) \(O^*(3.83^k)\)  
  [Cao, Chen, Liu ’10]
- \(\mathcal{F} = \{\theta_r\}\) \(O^*(c^k)\)  
  [Joret, Paul, S., Saurabh, Thomassé ’11]
- \(\mathcal{F} = \{K_4\}\) \(O^*(c^k)\)  
  [Kim, Paul, Philip ’12]

**General case:**

- **Planar-\(\mathcal{F}\)-Deletion** is **FPT**.  
  [Roberston and Seymour’s Graph Minors theory]
- \(2^{O(k \log k)} \cdot n^{O(1)}\) -time algorithm based on standard DP.  
  [Fomin, Lokshtanov, Misra, Saurabh ’11]
- \(2^{O(k \log k)} \cdot n^2\) -time algorithm.  
  [Fomin, Lokshtanov, Misra, Saurabh ’11]
- \(2^{O(k)} \cdot n \log^2 n\) -time algorithm for **Planar-Connected-\(\mathcal{F}\)-Deletion**.  
  [Fomin, Lokshtanov, Misra, Saurabh ’12]
Our result

**Theorem**

The **PLANAR-$\mathcal{F}$-Deletion** problem can be solved in time $2^{O(k)} \cdot n^2$.

- This result unifies a number of algorithms in the literature.
Our result

**Theorem**

*The Planar-$\mathcal{F}$-Deletion problem can be solved in time* $2^{O(k)} \cdot n^2$.

- This result unifies a number of algorithms in the literature.

- No hope for a $2^{o(k)} \cdot n^{O(1)}$-time algorithm (under ETH). [Chen et al. '05]

That is, the function $2^{O(k)}$ in our theorem is best possible.
1 Preliminaries

2 Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3 Single-exponential algorithm for PLANAR-$\mathcal{F}$-DELETION
   - Motivation and our result
   - Sketch of proof
   - Further research

4 Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
First step: use iterative compression

Using iterative compression the Planar-$\mathcal{F}$-Deletion problem can be reduced in single-exponential time to the following problem:
First step: use iterative compression

Using iterative compression the Planar-$\mathcal{F}$-Deletion problem can be reduced in single-exponential time to the following problem:

**Disjoint Planar-$\mathcal{F}$-Deletion**

**Input:** A graph $G$, a non-negative integer $k$, and a set $X \subseteq V(G)$ with $|X| = k$ s.t. $G - X$ is $\mathcal{F}$-minor-free.
First step: use iterative compression

Using iterative compression the Planar-$\mathcal{F}$-Deletion problem can be reduced in single-exponential time to the following problem:

**Disjoint Planar-$\mathcal{F}$-Deletion**

**Input:** A graph $G$, a non-negative integer $k$, and a set $X \subseteq V(G)$ with $|X| = k$ s.t. $G - X$ is $\mathcal{F}$-minor-free.

**Parameter:** The integer $k$.

**Question:** Does $G$ have a set $\tilde{X} \subseteq V(G) \setminus X$ such that $|\tilde{X}| < k$ and $G - \tilde{X}$ is $H$-minor-free for every $H \in \mathcal{F}$?

We call $\tilde{X}$ an alternative solution.
First step: use iterative compression

Using iterative compression the Planar-$\mathcal{F}$-Deletion problem can be reduced in single-exponential time to the following problem:

**Disjoint Planar-$\mathcal{F}$-Deletion**

**Input:** A graph $G$, a non-negative integer $k$, and a set $X \subseteq V(G)$ with $|X| = k$ s.t. $G - X$ is $\mathcal{F}$-minor-free.

**Parameter:** The integer $k$.

**Question:** Does $G$ have a set $\tilde{X} \subseteq V(G) \setminus X$ such that $|\tilde{X}| < k$ and $G - \tilde{X}$ is $H$-minor-free for every $H \in \mathcal{F}$?

We call $\tilde{X}$ an alternative solution.

**Lemma (well-known)**

*If **Disjoint Planar-$\mathcal{F}$-Deletion** can be solved in time $O^*(c^k)$ for some $c \in \mathbb{N}^+$, then **Planar-$\mathcal{F}$-Deletion** can be solved in $O^*((c + 1)^k)$.*
Working hypothesis: an alternative solution \( \tilde{X} \) does exist in \( G - X \).
Working hypothesis: an alternative solution $\tilde{X}$ does exist in $G - X$.

Observation:
If $(G, X, k)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then
- $G[X]$ is $\mathcal{F}$-minor-free
- $G[V \setminus X]$ is $\mathcal{F}$-minor-free
Working hypothesis: an alternative solution $\tilde{X}$ does exist in $G - X$.

Observation:
If $(G, X, k)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then

- $G[X]$ is $\mathcal{F}$-minor-free $\Rightarrow$ $G[X]$ has bounded $\text{tw}!!$
- $G[V \setminus X]$ is $\mathcal{F}$-minor-free $\Rightarrow$ $G[V \setminus X]$ has bounded $\text{tw}!!$
Working hypothesis: an alternative solution $\tilde{X}$ does exist in $G - X$.

Observation: If $(G, X, k)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then

- $G[X]$ is $\mathcal{F}$-minor-free $\Rightarrow$ $G[X]$ has bounded tw!!
- $G[V \setminus X]$ is $\mathcal{F}$-minor-free $\Rightarrow$ $G[V \setminus X]$ has bounded tw!!

☆ Let $r := |V(H)|$ for $H$ being some planar graph in the family $\mathcal{F}$.
Working hypothesis: an alternative solution $\tilde{X}$ does exist in $G - X$.

Observation:
If $(G, X, k)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then
- $G[X]$ is $\mathcal{F}$-minor-free $\Rightarrow$ $G[X]$ has bounded tw!!
- $G[V \setminus X]$ is $\mathcal{F}$-minor-free $\Rightarrow$ $G[V \setminus X]$ has bounded tw!!

\[ \star \] Let $r := |V(H)|$ for $H$ being some planar graph in the family $\mathcal{F}$.

\[ \star \] A connected component $C$ of $G - X$ is called a bloom component if $|N_X(C)| \geq r$, and a bud component otherwise.
Recall that a $\beta$-protrusion in a graph $G$ is a subset $Y \subseteq V(G)$ such that $|\partial(Y)| \leq \beta$ and $\text{tw}(G[Y]) \leq \beta$. 
Recall that a $\beta$-protrusion in a graph $G$ is a subset $Y \subseteq V(G)$ such that $|\partial(Y)| \leq \beta$ and $\text{tw}(G[Y]) \leq \beta$.

A partition $\mathcal{P} = Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ with $\max\{\ell, |Y_0|\} \leq \alpha$ is an $(\alpha, \beta)$-protrusion decomposition if for every $1 \leq i \leq \ell$,

$$N(Y_i) \subseteq Y_0 \quad \text{and} \quad Y_i \cup N_{Y_0}(Y_i) \text{ is a } \beta\text{-protrusion}$$
Recall that a $\beta$-protrusion in a graph $G$ is a subset $Y \subseteq V(G)$ such that $|\partial(Y)| \leq \beta$ and $\text{tw}(G[Y]) \leq \beta$.

A partition $\mathcal{P} = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell$ of $V(G)$ with $\max\{\ell, |Y_0|\} \leq \alpha$ is an $(\alpha, \beta)$-protrusion decomposition if for every $1 \leq i \leq \ell$,

$$N(Y_i) \subseteq Y_0 \quad \text{and} \quad Y_i \cup N_{Y_0}(Y_i) \text{ is a } \beta\text{-protrusion}$$

$\mathcal{P}$ is linear with respect to a parameter $k$ whenever $\alpha = O(k)$. 
We will use our algorithm to compute protrusion decompositions.
Algorithm to solve **DISJOINT PLANAR-$\mathcal{F}$-DELETION**

☆ Recall that $r = |V(H)|$, 

☆ But it turns out that, with input $(G, X, r)$, the set $Y_0$ output by our algorithm does not define a linear protrusion decomposition of $G$, which is crucial for us...

1. Guess the intersection $I = \tilde{X} \cap Y_0$ of the alt. solution $\tilde{X}$ with $Y_0$ s.t.:
   
   $G - I$ has a linear protrusion decomposition $P = Y_0 \sqcup Y_1 \sqcup \cdots \sqcup Y_\ell$ with $X \subseteq Y_0$ and $\tilde{X} \setminus I \subseteq V(G) \setminus Y_0$.

2. By carefully analyzing the output of our bag marking algorithm, finally compute $\tilde{X} \setminus I$, given a linear protrusion decomposition.

Both steps can be done in single-exponential time.
Recall that $r = |V(H)|$, and that $\text{tw}(G[V \setminus X]) \leq t_F$,
Recall that $r = |V(H)|$, and that $\text{tw}(G[V \setminus X]) \leq t_F$, so the set $X \subseteq V(G)$ will be the \text{treewidth-bounding set} which is given to the algorithm.
Algorithm to solve **Disjoint Planar-\(\mathcal{F}\)-Deletion**

- Recall that \(r = |V(H)|\), and that \(\text{tw}(G[V \setminus X]) \leq t_\mathcal{F}\), so the set \(X \subseteq V(G)\) will be the treewidth-bounding set which is given to the algorithm.

- But it turns out that, with input \((G, X, r)\), the set \(Y_0\) output by our algorithm does not define a linear protrusion decomposition of \(G\), which is crucial for us...
Algorithm to solve **Disjoint Planar-\(\mathcal{F}\)-Deletion**

- Recall that \(r = |V(H)|\), and that \(\text{tw}(G[V \setminus X]) \leq t_{\mathcal{F}}\), so the set \(X \subseteq V(G)\) will be the treewidth-bounding set which is given to the algorithm.

- But it turns out that, with input \((G, X, r)\), the set \(Y_0\) output by our algorithm does not define a linear protrusion decomposition of \(G\), which is crucial for us...

1. **Guess the intersection** \(I = \tilde{X} \cap Y_0\) of the alt. solution \(\tilde{X}\) with \(Y_0\) s.t.:
   - \(G - I\) has a linear protrusion decomposition
     \[\mathcal{P} = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell\]
   - with \(X \subseteq Y_0\) and \(\tilde{X} \setminus I \subseteq V(G) \setminus Y_0\).
Algorithm to solve **Disjoint Planar-$\mathcal{F}$-Deletion**

- Recall that $r = |V(H)|$, and that $\text{tw}(G[V \setminus X]) \leq t_\mathcal{F}$, so the set $X \subseteq V(G)$ will be the treewidth-bounding set which is given to the algorithm.

- But it turns out that, with input $(G, X, r)$, the set $Y_0$ output by our algorithm does not define a linear protrusion decomposition of $G$, which is crucial for us...

1. Guess the intersection $I = \tilde{X} \cap Y_0$ of the alt. solution $\tilde{X}$ with $Y_0$ s.t.:
   - $G - I$ has a linear protrusion decomposition
     $$\mathcal{P} = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell$$
   - with $X \subseteq Y_0$ and $\tilde{X} \setminus I \subseteq V(G) \setminus Y_0$.

By carefully analyzing the output of our bag marking algorithm
Algorithm to solve **Disjoint Planar-\(\mathcal{F}\)-Deletion**

★ Recall that \( r = |V(H)| \), and that \( \text{tw}(G[V \setminus X]) \leq t_\mathcal{F} \), so the set \( X \subseteq V(G) \) will be the **treewidth-bounding set** which is given to the algorithm.

★ But it turns out that, with input \((G, X, r)\), the set \( Y_0 \) output by our algorithm does **not** define a **linear** protrusion decomposition of \( G \), which is crucial for us...

1. **Guess the intersection** \( I = \tilde{X} \cap Y_0 \) of the alt. solution \( \tilde{X} \) with \( Y_0 \) s.t.:
   - \( G - I \) has a **linear** protrusion decomposition
     \[ \mathcal{P} = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell \]
   - with \( X \subseteq Y_0 \) and \( \tilde{X} \setminus I \subseteq V(G) \setminus Y_0 \).

   **By carefully analyzing the output of our bag marking algorithm**

2. **Finally, compute** \( \tilde{X} \setminus I \), given a linear protrusion decomposition.
**Algorithm to solve** **Disjoint Planar-\(\mathcal{F}\)-Deletion**

召回\(r = |V(H)|\)，且\(\text{tw}(G[V \setminus X]) \leq t_{\mathcal{F}}\)，所以整个集\(X \subseteq V(G)\)将作为treewidth-bounding set给算法。

但事实证明，给定输入\((G, X, r)\)，由我们算法输出的集\(Y_0\)不定义线性突起分解，这对我们至关重要。

1. **Guess the intersection** \(I = \tilde{X} \cap Y_0\) of the alt. solution \(\tilde{X}\) with \(Y_0\) s.t.: \(G - I\) has a linear protrusion decomposition \(\mathcal{P} = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell\) with \(X \subseteq Y_0\) and \(\tilde{X} \setminus I \subseteq V(G) \setminus Y_0\). By carefully analyzing the output of our bag marking algorithm.

2. Finally, compute \(\tilde{X} \setminus I\), given a linear protrusion decomposition. Based on the finite index of MSO-definable properties (automaton theory).
Algorithm to solve **Disjoint Planar-\(\mathcal{F}\)-Deletion**

- Recall that \(r = |V(H)|\), and that \(\text{tw}(G[V \setminus X]) \leq t_{\mathcal{F}}\), so the set \(X \subseteq V(G)\) will be the treewidth-bounding set which is given to the algorithm.

- But it turns out that, with input \((G, X, r)\), the set \(Y_0\) output by our algorithm does not define a linear protrusion decomposition of \(G\), which is crucial for us...

1. **Guess the intersection** \(I = \tilde{X} \cap Y_0\) of the alt. solution \(\tilde{X}\) with \(Y_0\) s.t.:
   - \(G - I\) has a linear protrusion decomposition
     \[ P = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell \]
   - with \(X \subseteq Y_0\) and \(\tilde{X} \setminus I \subseteq V(G) \setminus Y_0\).

   **By carefully analyzing the output of our bag marking algorithm**

2. **Finally, compute** \(\tilde{X} \setminus I\), given a linear protrusion decomposition.

   **Based on the finite index of MSO-definable properties (automaton theory)**

- Both steps can be done in single-exponential time.
First step: analysis of the bag marking algorithm

Lemma (edge simulation to chop bloom components)

If $C_1, \ldots, C_\ell$ is a collection of connected pairwise vertex-disjoint subgraphs of $G - X$ such that $|N_X(C_i)| \geq r$ for $1 \leq i \leq \ell$, then $\ell \leq (1 + \alpha_r) \cdot k$. 
First step: analysis of the bag marking algorithm

Lemma (edge simulation to chop bloom components)

If \( C_1, \ldots, C_\ell \) is a collection of connected pairwise vertex-disjoint subgraphs of \( G - X \) such that \( |N_X(C_i)| \geq r \) for \( 1 \leq i \leq \ell \), then \( \ell \leq (1 + \alpha_r) \cdot k \).

Proposition (Thomason ’01)

There exists a constant \( \alpha < 0.320 \) such that any \( n \)-vertex graph with no \( K_r \)-minor has at most \( \alpha_r \cdot n = (\alpha \cdot r \sqrt{\log r}) \cdot n \) edges.

(Recall that \( r = |V(H)| \), for \( H \) being any planar graph in \( \mathcal{F} \))
First step: analysis of the bag marking algorithm

Lemma (edge simulation to chop bloom components)
If $C_1, \ldots, C_\ell$ is a collection of connected pairwise vertex-disjoint subgraphs of $G - X$ such that $|N_X(C_i)| \geq r$ for $1 \leq i \leq \ell$, then $\ell \leq (1 + \alpha r) \cdot k$.

Proposition (Thomason ’01)
There exists a constant $\alpha < 0.320$ such that any $n$-vertex graph with no $K_r$-minor has at most $\alpha_r \cdot n = (\alpha \cdot r \sqrt{\log r}) \cdot n$ edges.

(Recall that $r = |V(H)|$, for $H$ being any planar graph in $\mathcal{F}$)
First step: analysis of the bag marking algorithm

**Lemma (edge simulation to chop bloom components)**

If $C_1, \ldots, C_\ell$ is a collection of connected pairwise vertex-disjoint subgraphs of $G - X$ such that $|N_X(C_i)| \geq r$ for $1 \leq i \leq \ell$, then $\ell \leq (1 + \alpha_r) \cdot k$.

**Proposition (Thomason ’01)**

There exists a constant $\alpha < 0.320$ such that any $n$-vertex graph with no $K_r$-minor has at most $\alpha_r \cdot n = (\alpha \cdot r \sqrt{\log r}) \cdot n$ edges.

(Recall that $r = |V(H)|$, for $H$ being any planar graph in $\mathcal{F}$)
Consider an optimal tree-decomposition $\mathcal{T} = (T, B)$ of a “bloom” connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$)
Chopping bloom components (2)

Consider an optimal tree-decomposition $\mathcal{T} = (T, B)$ of a “bloom” connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$)

Recall our bottom-up Bag Marking algorithm:
if a bag $B$ is the LCA of two marked bags of $M$, or $G_B$ contains a connected bloom component, then

$\mathcal{M} \leftarrow \mathcal{M} \cup \{B\}$ and remove the vertices in $B$ from the bags of $\mathcal{T}$
Chopping bloom components (2)

Consider an optimal tree-decomposition $\mathcal{T} = (T, B)$ of a “bloom” connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$)

Recall our bottom-up Bag Marking algorithm:

if a bag $B$ is the LCA of two marked bags of $\mathcal{M}$, or $G_B$ contains a connected bloom component, then

$\mathcal{M} \leftarrow \mathcal{M} \cup \{B\}$ and remove the vertices in $B$ from the bags of $\mathcal{T}$
Consider an optimal tree-decomposition $\mathcal{T} = (T, B)$ of a "bloom" connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$).

Recall our bottom-up Bag Marking algorithm:
if a bag $B$ is the LCA of two marked bags of $M$, or $G_B$ contains a connected bloom component, then

- $M \leftarrow M \cup \{B\}$ and remove the vertices in $B$ from the bags of $T$.
Consider an optimal tree-decomposition $T = (T, B)$ of a “bloom” connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$).

Recall our bottom-up Bag Marking algorithm:

If a bag $B$ is the LCA of two marked bags of $M$, or $G_B$ contains a connected bloom component, then

- $M \leftarrow M \cup \{B\}$ and remove the vertices in $B$ from the bags of $T$.
Chopping bloom components (2)

Consider an optimal tree-decomposition $\mathcal{T} = (T, B)$ of a “bloom” connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$)

Recall our bottom-up Bag Marking algorithm:

if a bag $B$ is the LCA of two marked bags of $M$, or 

$G_B$ contains a connected bloom component, then

$\mathcal{M} \leftarrow \mathcal{M} \cup \{B\}$ and remove the vertices in $B$ from the bags of $\mathcal{T}$
Consider an optimal tree-decomposition $\mathcal{T} = (T, B)$ of a “bloom” connected component $C$ of $G - X$ (i.e., $|N_X(C)| \geq r$)

Recall our bottom-up Bag Marking algorithm:

if a bag $B$ is the LCA of two marked bags of $\mathcal{M}$, or $G_B$ contains a connected bloom component, then

- $\mathcal{M} \leftarrow \mathcal{M} \cup \{B\}$ and remove the vertices in $B$ from the bags of $\mathcal{T}$
Lemma ($|Y_0| = O(k)$ and every component is a protrusion)

If $(G, X, k)$ is a `Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion`, then

- $Y_0 = X \cup V(M)$ has size at most $k + 2t_\mathcal{F} \cdot (1 + \alpha_r) \cdot k$.
- Every connected component $C$ of $G - Y_0$ satisfies
  
  $|N_X(C)| \leq r$ and $|N_{Y_0}(C)| \leq r + 2t_\mathcal{F}$. 

Note that $k = |X|$, $tw(G - X) \leq t_\mathcal{F}$, and $|M| \leq (1 + \alpha_r) \cdot k$ (by the "edge simulation" Lemma).
Lemma ($|Y_0| = O(k)$ and every component is a protrusion)

If $(G, X, k)$ is a Yes-instance of **Disjoint Planar-\mathcal{F}-Deletion**, then

- $Y_0 = X \cup V(\mathcal{M})$ has size at most $k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k$.
- Every connected component $C$ of $G - Y_0$ satisfies $|N_X(C)| \leq r$ and $|N_{Y_0}(C)| \leq r + 2t_{\mathcal{F}}$.
- Note that $k = |X|$.
Lemma ($|Y_0| = O(k)$ and every component is a protrusion)

If $(G, X, k)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then

- $Y_0 = X \cup V(\mathcal{M})$ has size at most $k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k$.
- Every connected component $C$ of $G - Y_0$ satisfies
  
  $|N_X(C)| \leq r$ and $|N_{Y_0}(C)| \leq r + 2t_{\mathcal{F}}$.

- Note that $k = |X|$, 
- $\text{tw}(G - X) \leq t_{\mathcal{F}}$, and
Lemma ($|Y_0| = O(k)$ and every component is a protrusion)

If $(G, X, k)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then

1. $Y_0 = X \cup V(M)$ has size at most $k + 2t_{\mathcal{F}} \cdot (1 + \alpha_r) \cdot k$.
2. Every connected component $C$ of $G - Y_0$ satisfies
   $$|N_X(C)| \leq r \quad \text{and} \quad |N_{Y_0}(C)| \leq r + 2t_{\mathcal{F}}.$$

Note that $k = |X|$, $tw(G - X) \leq t_{\mathcal{F}}$, and $|M| \leq (1 + \alpha_r) \cdot k$ (by the “edge simulation” Lemma).
Remark: Therefore, $Y_0$ and the connected components of $G - Y_0$ form a protrusion decomposition of $G$... but not a linear one!
Computing a linear protrusion decomposition

Remark: Therefore, $Y_0$ and the connected components of $G - Y_0$ form a protrusion decomposition of $G$... but not a linear one!

We need that $\#\text{protrusions} = O(k)$. 

Remark: Therefore, $Y_0$ and the connected components of $G - Y_0$ form a protrusion decomposition of $G$... but not a linear one!

We need that $\#\text{protrusions} = O(k)$.

Branching step:

Guess $I = \tilde{X} \cap Y_0$ among the $2^{O(k)}$ subsets of $V(\mathcal{M})$.
Computing a linear protrusion decomposition

Remark: Therefore, $Y_0$ and the connected components of $G - Y_0$ form a protrusion decomposition of $G$... but not a linear one!

We need that $\#\text{protrusions} = O(k)$.

Branching step:

Guess $I = \tilde{X} \cap Y_0$ among the $2^{O(k)}$ subsets of $V(M)$

Let $G_I := G - I$. Recall that a cluster of $G_I - Y_0$ is a maximal set of connected components of $G_I - Y_0$ with the same neighborhood in $Y_0$. 
Linear protrusion decomposition (2)

Lemma (For some choice of $I$, $\#\text{clusters} = O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then the number $\ell$ of clusters of $G_I - Y_0$ is at most $(5t_\mathcal{F} \alpha_r \mu_r) \cdot k$. 
Linear protrusion decomposition (2)

Lemma (For some choice of $I$, $\#\text{clusters} = O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a $\text{Yes}$-instance of $\text{Disjoint Planar-$\mathcal{F}$-Deletion}$, then the number $\ell$ of clusters of of $G_I - Y_0$ is at most $(5t_F \alpha_r \mu_r) \cdot k$.

Proposition (Fomin, Oum, Thilikos ’10)

There exists a constant $\mu < 11.355$ such that for all $r > 2$, every $n$-vertex graph with no $K_r$-minor has at most $\mu_r \cdot n = 2^{\mu \cdot r \log \log r} \cdot n$ cliques.
Linear protrusion decomposition (2)

Lemma (For some choice of $I$, $\#\text{clusters} = O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a \textsc{Yes}-instance of \textsc{Disjoint Planar-$\mathcal{F}$-Deletion}, then the number $\ell$ of clusters of $G_I - Y_0$ is at most $(5t_{\mathcal{F}} \alpha_r \mu_r) \cdot k$.

Proposition (Fomin, Oum, Thilikos ’10)

There exists a constant $\mu < 11.355$ such that for all $r > 2$, every $n$-vertex graph with no $K_r$-minor has at most $\mu_r \cdot n = 2^{\mu_r \log \log r} \cdot n$ cliques.

\* At most $\ell' = k - |I|$ clusters $C_1, \ldots, C_{\ell'}$ intersect the alternative solution $\tilde{X}$. 
Linear protrusion decomposition (2)

Lemma (For some choice of $I$, \( \#\text{clusters} = O(k) \))

If \((G_I, Y_0 \setminus I, k - |I|)\) is a Yes-instance of Disjoint Planar-\(\mathcal{F}\)-Deletion, then the number \(\ell\) of clusters of of \(G_I - Y_0\) is at most \((5t_F \alpha_r \mu_r) \cdot k\).

Proposition (Fomin, Oum, Thilikos ’10)

There exists a constant \(\mu < 11.355\) such that for all \(r > 2\), every \(n\)-vertex graph with no \(K_r\)-minor has at most \(\mu_r \cdot n = 2^{\mu \cdot r \log \log r} \cdot n\) cliques.

We have that \(G' = G_I - \bigcup_{i=1}^{\ell'} C_i\) is \(\mathcal{F}\)-minor-free.
Lemma (For some choice of $I$, #clusters $= O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then the number $\ell$ of clusters of $G_I - Y_0$ is at most $(5t_{\mathcal{F}} \alpha_r \mu_r) \cdot k$.

Proposition (Fomin, Oum, Thilikos ’10)

There exists a constant $\mu < 11.355$ such that for all $r > 2$, every $n$-vertex graph with no $K_r$-minor has at most $\mu_r \cdot n = 2^{\mu \cdot r \log \log r} \cdot n$ cliques.

☆ Using edge simulation we construct a minor of $G'$ on vertices of $Y_0$. 

Linear protrusion decomposition (2)
Lemma (For some choice of $I$, $\#\text{clusters} = O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then the number $\ell$ of clusters of of $G_I - Y_0$ is at most $(5t_\mathcal{F}\alpha_r\mu_r) \cdot k$.

Proposition (Fomin, Oum, Thilikos ’10)

There exists a constant $\mu < 11.355$ such that for all $r > 2$, every $n$-vertex graph with no $K_r$-minor has at most $\mu_r \cdot n = 2^{\mu_r \log \log r} \cdot n$ cliques.

⋆ As before, the number of clusters used so far is at most $\alpha_r \cdot k$. 
**Lemma** (For some choice of $I$, \#clusters $= O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a **Yes-instance** of **Disjoint Planar-$\mathcal{F}$-Deletion**, then the **number $\ell$ of clusters** of of $G_I - Y_0$ is at most $\left(5t_F \alpha_r \mu_r\right) \cdot k$.

**Proposition** (Fomin, Oum, Thilikos ’10)

There exists a constant $\mu < 11.355$ such that for all $r > 2$, every $n$-vertex graph with no $K_r$-minor has at most $\mu_r \cdot n = 2^{\mu \cdot r \log \log r} \cdot n$ **cliques**.

★ When we cannot add more edges, all neighborhoods of clusters are **cliques**!
Lemma (For some choice of $I$, $\#\text{clusters} = O(k)$)

If $(G_I, Y_0 \setminus I, k - |I|)$ is a Yes-instance of Disjoint Planar-$\mathcal{F}$-Deletion, then the number $\ell$ of clusters of $G_I - Y_0$ is at most $(5t_\mathcal{F} \alpha_r \mu_r) \cdot k$.

Proposition (Fomin, Oum, Thilikos ’10)

There exists a constant $\mu < 11.355$ such that for all $r > 2$, every $n$-vertex graph with no $K_r$-minor has at most $\mu_r \cdot n = 2^{\mu \cdot r \log \log r} \cdot n$ cliques.

★ Now we use the Proposition: the number of remaining clusters is $\mu_r \cdot k$. 
Therefore, the partition $P = Y_0 \sqcup C_1 \sqcup \cdots \sqcup C_\ell$ is a $(O(k), r + 2t_F)$-protrusion decomposition of $G_I = G - I$. 

Recall the two main steps of our algorithm:

1. Guess the intersection $I = \tilde{X} \cap Y_0$ of the alt. solution $\tilde{X}$ with $Y_0$ s.t.:
   - $G - I$ has a linear protrusion decomposition $P = Y_0 \sqcup C_1 \sqcup \cdots \sqcup C_\ell$ with $X \subseteq Y_0$ and $\tilde{X} \setminus I \subseteq V(G) \setminus Y_0$.
2. Finally, compute $\tilde{X} \setminus I$, given a linear protrusion decomposition.
Back to the road map of the algorithm

Therefore, the partition $\mathcal{P} = Y_0 \uplus C_1 \uplus \cdots \uplus C_\ell$ is a $(O(k), r + 2t_\mathcal{F})$-protrusion decomposition of $G_I = G - I$.

Recall the two main steps of our algorithm:

1. Guess the intersection $I = \tilde{X} \cap Y_0$ of the alt. solution $\tilde{X}$ with $Y_0$ s.t.:
   - $G - I$ has a linear protrusion decomposition
     $\mathcal{P} = Y_0 \uplus C_1 \uplus \cdots \uplus C_\ell$
   - $X \subseteq Y_0$ and $\tilde{X} \setminus I \subseteq V(G) \setminus Y_0$. 

2. Finally, compute $\tilde{X} \setminus I$, given a linear protrusion decomposition.
Back to the road map of the algorithm

Therefore, the partition $\mathcal{P} = Y_0 \uplus C_1 \uplus \cdots \uplus C_\ell$ is a

$$(O(k), r + 2t_F)$$-protrusion decomposition of $G_I = G - I$.

Recall the two main steps of our algorithm:

1. Guess the intersection $I = \tilde{X} \cap Y_0$ of the alt. solution $\tilde{X}$ with $Y_0$ s.t.:
   - $G - I$ has a linear protrusion decomposition
     $$\mathcal{P} = Y_0 \uplus C_1 \uplus \cdots \uplus C_\ell$$
     with $X \subseteq Y_0$ and $\tilde{X} \setminus I \subseteq V(G) \setminus Y_0$.

2. Finally, compute $\tilde{X} \setminus I$, given a linear protrusion decomposition.

Based on the finite index of MSO-definable properties (automaton theory)
Solving the problem when given a linear protrusion decomposition

Main ingredients of our approach:

⋆ We define an equivalence relation on subsets of vertices of each restricted protrusion $Y_i$ (roughly, same class if they behave in the same way).

⋆ Each of these equiv. relations defines finitely many equivalence classes s.t. any partial solution on $Y_i$ can be replaced with one of the representatives. (by the finite index of MSO-definable properties) [Bodlaender, de Fluiter '01]

⋆ We use a decomposability property of the solution: there exists a solution which is formed by the union of one representative per restricted protrusion.

⋆ To make the algorithm constructive and uniform on the family $F$, we use classic arguments from tree automaton theory (like method of test sets).
Solving the problem when given a linear protrusion decomposition

Main ingredients of our approach:

- We define an equivalence relation on subsets of vertices of each restricted protrusion $Y_i$ (roughly, same class if they behave in the same way).

- Each of these equivalence relations define finitely many equivalence classes such that any partial solution on $Y_i$ can be replaced with one of the representatives (by the finite index of MSO-definable properties).

- We use a decomposability property of the solution: there exists a solution which is formed by the union of one representative per restricted protrusion.

- To make the algorithm constructive and uniform on the family $F$, we use classic arguments from tree automaton theory (like method of test sets).
Solving the problem when given a linear protrusion decomposition

Main ingredients of our approach:

★ We define an equivalence relation on subsets of vertices of each restricted protrusion $Y_i$ (roughly, same class if they behave in the same way).

★ Each of these equiv. relations defines finitely many equivalence classes s.t. any partial solution on $Y_i$ can be replaced with one of the representatives. (by the finite index of MSO-definable properties) [Bodlaender, de Fluiter ’01]
Main ingredients of our approach:

- We define an equivalence relation on subsets of vertices of each restricted protrusion $Y_i$ (roughly, same class if they behave in the same way).
- Each of these equiv. relations defines finitely many equivalence classes s.t. any partial solution on $Y_i$ can be replaced with one of the representatives. (by the finite index of MSO-definable properties) [Bodlaender, de Fluiter '01]
- We use a decomposability property of the solution: there exists a solution which is formed by the union of one representative per restricted protrusion.
Solving the problem when given a linear protrusion decomposition

Main ingredients of our approach:

- We define an equivalence relation on subsets of vertices of each restricted protrusion $Y_i$ (roughly, same class if they behave in the same way).
- Each of these equivalence relations defines finitely many equivalence classes such that any partial solution on $Y_i$ can be replaced with one of the representatives. (by the finite index of MSO-definable properties) [Bodlaender, de Fluiter '01]
- We use a decomposability property of the solution: there exists a solution which is formed by the union of one representative per restricted protrusion.
- To make the algorithm constructive and uniform on the family $\mathcal{F}$, we use classic arguments from tree automaton theory (like method of test sets).
1. Preliminaries

2. Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3. Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   - Motivation and our result
   - Sketch of proof
   - Further research

4. Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
Conclusions and further research

Theorem

The **Planar-\(\mathcal{F}\)-Deletion** problem can be solved in time \(2^{O(k)} \cdot n^2\).
Theorem

The **Planar-\(F\)-Deletion** problem can be solved in time \(2^{O(k)} \cdot n^2\).

★ Can a single-exponential algorithm exist when the family \(F\) does not contain any planar graph?

For \(F = \{K_5, K_{3,3}\}\), an explicit FPT algorithm is known. It runs in time \(2^{O(k \log k)} \cdot n\).

[Jansen, Lokshtanov, Saurabh '14]
Conclusions and further research

Theorem

The **Planar-\(\mathcal{F}\)-Deletion** problem can be solved in time \(2^{O(k)} \cdot n^2\).

★ Can a single-exponential algorithm exist when the family \(\mathcal{F}\) does not contain any planar graph?

For \(\mathcal{F} = \{K_5, K_{3,3}\}\), an explicit FPT algorithm is known. It runs in time \(2^{O(k \log k)} \cdot n\).

[Jansen, Lokshtanov, Saurabh '14]

★ There exists a randomized constant-factor approximation algorithm for **Planar-\(\mathcal{F}\)-Deletion**.

Finding a deterministic constant-factor approximation remains open.

[Fomin, Lokshtanov, Misra, Saurabh '12]
Conclusions and further research

Theorem

The Planar-$\mathcal{F}$-Deletion problem can be solved in time $2^{O(k)} \cdot n^2$.

- Can a single-exponential algorithm exist when the family $\mathcal{F}$ does not contain any planar graph?

  For $\mathcal{F} = \{K_5, K_{3,3}\}$, an explicit FPT algorithm is known. It runs in time $2^{O(k \log k)} \cdot n$.

  [Jansen, Lokshtanov, Saurabh '14]

- There exists a randomized constant-factor approximation algorithm for Planar-$\mathcal{F}$-Deletion.

  Finding a deterministic constant-factor approximation remains open.

  [Fomin, Lokshtanov, Misra, Saurabh '12]

- We could forbid the family of graphs $\mathcal{F}$ according to another containment relation, like topological minor.
1 Preliminaries

2 Protrusion decompositions
   • Definitions
   • A simple algorithm to compute them

3 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   • Motivation and our result
   • Sketch of proof
   • Further research

4 Linear kernels on graphs without topological minors
   • Motivation and our result
   • Idea of proof
   • Further research
1. Preliminaries

2. Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3. Single-exponential algorithm for \textsc{Planar-$\mathcal{F}$-Deletion}
   - Motivation and our result
   - Sketch of proof
   - Further research

4. Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
A kernel for a parameterized problem \( \Pi \) is an algorithm that given \((x, k)\) outputs, in time polynomial in \(|x| + k\), an instance \((x', k')\) s.t.:

- \((x, k) \in \Pi \) if and only if \((x', k') \in \Pi\), and

<table>
<thead>
<tr>
<th>Function (g)</th>
<th>Description</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(1))</td>
<td>Polynomial kernel</td>
<td>(g(k) \in \mathbb{O}(k))</td>
</tr>
<tr>
<td>(O(k))</td>
<td>Linear kernel</td>
<td>(g(k) \in \mathbb{O}(k^2))</td>
</tr>
</tbody>
</table>

Folklore result: for a parameterized problem \( \Pi \), \( \Pi \) is FPT \iff \( \Pi \) admits a kernel.

Question: which FPT problems admit linear or polynomial kernels?
Kernels

- A **kernel** for a parameterized problem Π is an algorithm that given \((x, k)\) outputs, in time polynomial in \(|x| + k\), an instance \((x', k')\) s.t.:
  - \((x, k) \in \Pi\) if and only if \((x', k') \in \Pi\), and
  - Both \(|x'|, k' \leq g(k)|\), where \(g\) is some computable function.

The function \(g\) is called the size of the kernel.

- If \(g(k) = O(1)\): Π admits a polynomial kernel.
- If \(g(k) = O(k)\): Π admits a linear kernel.

Folklore result: for a parameterized problem Π, \(\Pi\) is FPT \(\iff\) Π admits a kernel.
A kernel for a parameterized problem $\Pi$ is an algorithm that given $(x, k)$ outputs, in time polynomial in $|x| + k$, an instance $(x', k')$ s.t.:

- $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and
- Both $|x'|, k' \leq g(k)$, where $g$ is some computable function.

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

- If $g(k) = k^{O(1)}$: $\Pi$ admits a polynomial kernel.
- If $g(k) = O(k)$: $\Pi$ admits a linear kernel.
Kernels

- A kernel for a parameterized problem $\Pi$ is an algorithm that given $(x, k)$ outputs, in time polynomial in $|x| + k$, an instance $(x', k')$ s.t.:
  - $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and
  - Both $|x'|, k' \leq g(k)$, where $g$ is some computable function.

- The function $g$ is called the size of the kernel.
  - If $g(k) = k^{O(1)}$: $\Pi$ admits a polynomial kernel.
  - If $g(k) = O(k)$: $\Pi$ admits a linear kernel.

- Folklore result: for a parameterized problem $\Pi$,
  
  $\Pi$ is FPT $\iff$ $\Pi$ admits a kernel
A kernel for a parameterized problem $\Pi$ is an algorithm that given $(x, k)$ outputs, in time \textit{polynomial} in $|x| + k$, an instance $(x', k')$ s.t.:

\begin{itemize}
  \item $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and
  \item Both $|x'|, k' \leq g(k)$, where $g$ is some computable function.
\end{itemize}

The function $g$ is called the \textit{size} of the kernel.

\begin{itemize}
  \item If $g(k) = k^{O(1)}$: $\Pi$ admits a \textit{polynomial} kernel.
  \item If $g(k) = O(k)$: $\Pi$ admits a \textit{linear} kernel.
\end{itemize}

Folklore result: for a parameterized problem $\Pi$,

\[ \Pi \text{ is FPT} \iff \Pi \text{ admits a kernel} \]

Question: which \textit{FPT} problems admit \textit{linear} or \textit{polynomial} kernels?
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.

- Therefore: $H$ minor of $G \Rightarrow H$ topological minor of $G$.

- Fixed $H$: $H$-minor-free graphs \( \subseteq \) $H$-topological-minor-free graphs.
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: $H$ minor of $G \Rightarrow H$ topological minor of $G$. 

Fixed $H$: $H$-minor-free graphs $\subseteq H$-topological-minor-free graphs.
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: $H$ minor of $G \iff H$ topological 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 deg $\leq 2$.

Therefore: \[ H \text{ minor of } G \not\equiv H \text{ topological minor of } G. \]

Fixed **H**: \[ H\text{-minor-free graphs} \subseteq H\text{-topological-minor-free graphs}. \]
Dominating Set on planar graphs. [Alber, Fellows, Niedermeier '04]
Dominating Set on planar graphs. [Alber, Fellows, Niedermeier '04]

Framework for several problems on planar graphs. [Guo, Niedermeier '04]
Linear kernels on sparse graphs – an overview

- **Dominating Set** on planar graphs.  
  [Alber, Fellows, Niedermeier ’04]

- Framework for several problems on planar graphs.  
  [Guo, Niedermeier ’04]

- Meta-result for graphs of bounded genus.  
  [Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos ’09]
Linear kernels on sparse graphs – an overview

- **Dominating Set** on planar graphs. [Alber, Fellows, Niedermeier '04]

- Framework for several problems on **planar** graphs. [Guo, Niedermeier '04]

- Meta-result for graphs of **bounded genus**. [Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos '09]

- Meta-result for **$H$-minor-free** graphs. [Fomin, Lokshtanov, Saurabh, Thilikos '10]
Linear kernels on sparse graphs – an overview

- **Dominating Set** on planar graphs. [Alber, Fellows, Niedermeier '04]
- Framework for several problems on planar graphs. [Guo, Niedermeier '04]
- Meta-result for graphs of bounded genus. [Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos '09]
- Meta-result for $H$-minor-free graphs. [Fomin, Lokshtanov, Saurabh, Thilikos '10]
- Meta-result for $H$-topological-minor-free graphs. [Our result]
Our result

Theorem

Fix a graph $H$. Let $\Pi$ be a parameterized graph problem on the class of $H$-topological-minor-free graphs that is treewidth-bounding and has finite integer index (FII). Then $\Pi$ admits a linear kernel.
Our result

Theorem

Fix a graph $H$. Let $\Pi$ be a parameterized graph problem on the class of $H$-topological-minor-free graphs that is \textit{treewidth-bounding} and has finite integer index (FII). Then $\Pi$ admits a linear kernel.

- A parameterized graph problem $\Pi$ is \textit{treewidth-bounding} if $\exists$ constants $c, t$ such that if $(G, k) \in \Pi$ then

\[ \exists X \subseteq V(G) \text{ s.t. } |X| \leq c \cdot k \text{ and } tw(G - X) \leq t. \]
Our result

**Theorem**

Fix a graph $H$. Let $\Pi$ be a parameterized graph problem on the class of $H$-topological-minor-free graphs that is treewidth-bounding and has finite integer index (FII). Then $\Pi$ admits a linear kernel.

- A parameterized graph problem $\Pi$ is treewidth-bounding if $\exists$ constants $c, t$ such that if $(G, k) \in \Pi$ then
  \[ \exists X \subseteq V(G) \text{ s.t. } |X| \leq c \cdot k \text{ and } \text{tw}(G - X) \leq t. \]
- FII allows us to replace large protrusions by smaller gadgets...
**Theorem**

Fix a graph $H$. Let $\Pi$ be a parameterized graph problem on the class of $H$-topological-minor-free graphs that is treewidth-bounding and has finite integer index (FII). Then $\Pi$ admits a linear kernel.

- A parameterized graph problem $\Pi$ is treewidth-bounding if $\exists$ constants $c$, $t$ such that if $(G, k) \in \Pi$ then
  \[ \exists X \subseteq V(G) \text{ s.t. } |X| \leq c \cdot k \text{ and } \text{tw}(G - X) \leq t. \]

- FII allows us to replace large protrusions by smaller gadgets...

★ We assume that the **gadgets are given**. Our algorithm is non-uniform.
Our result

Theorem

Fix a graph $H$. Let $\Pi$ be a parameterized graph problem on the class of $H$-topological-minor-free graphs that is treewidth-bounding and has finite integer index (FII). Then $\Pi$ admits a linear kernel.

- A parameterized graph problem $\Pi$ is treewidth-bounding if $\exists$ constants $c, t$ such that if $(G, k) \in \Pi$ then
  \[ \exists X \subseteq V(G) \text{ s.t. } |X| \leq c \cdot k \text{ and } tw(G - X) \leq t. \]
- FII allows us to replace large protrusions by smaller gadgets...

★ We assume that the gadgets are given. Our algorithm is non-uniform.

Problems affected by our result:

*Treewidth-$t$ Vertex Deletion, Chordal Vertex Deletion, Interval Vertex Deletion, Edge Dominating Set, Feedback Vertex Set, Connected Vertex Cover,* …
Linear kernels on sparse graphs – the conditions

- \( H \)-topological-minor-free
- \( H \)-minor-free
- bounded genus
- planar

\begin{align*}
\cup \\
\cup \\
\cup \\
\cup
\end{align*}

- treewidth-bounding
- bidimensional, separation property
- quasi-compact
- “distance-property”

(Figure by Felix Reidl)
Are our conditions very restrictive?

We require FII + treewidth-bounding

Conditions on $H$-minor-free graphs: bidimensional + separation property.

But it holds that bidimensional + separation property $\implies$ treewidth-bounding.

Thus, our results imply the linear kernels of Fomin, Lokshtanov, Saurabh, Thilikos '10.
Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

Bidimensional + separation property $\Rightarrow$ treewidth-bounding

Thus, our results imply the linear kernels of Fomin, Lokshtanov, Saurabh, Thilikos '10.
Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.
- What about requiring the problems to be treewidth-bounding?
Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

- What about requiring the problems to be treewidth-bounding?

  Conditions on $H$-minor-free graphs:
  bidimensional + separation property.  

  [Fomin, Lokshtanov, Saurabh, Thilikos '10]
Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

- What about requiring the problems to be treewidth-bounding?

  Conditions on $H$-minor-free graphs:
  bidimensional + separation property.  
  
  But it holds that

  bidimensional + separation property $\Rightarrow$ treewidth-bounding
Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

- What about requiring the problems to be treewidth-bounding?

Conditions on $H$-minor-free graphs:
bidimensional + separation property.  

[Fomin, Lokshtanov, Saurabh, Thilikos '10]

But it holds that

$$ \text{bidimensional} + \text{separation property} \Rightarrow \text{treewidth-bounding} $$

- Thus, our results imply the linear kernels of  

[Fomin, Lokshtanov, Saurabh, Thilikos '10]
1 Preliminaries

2 Protrusion decompositions
   - Definitions
   - A simple algorithm to compute them

3 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   - Motivation and our result
   - Sketch of proof
   - Further research

4 Linear kernels on graphs without topological minors
   - Motivation and our result
   - Idea of proof
   - Further research
Let $\Pi$ be a parameterized graph problem restricted to a graph class $G$ and let $G_1, G_2$ be two $t$-boundaried graphs in $G_t$. We say that $G_1 \equiv_{\Pi, t} G_2$ if there exists a constant $\Delta_{\Pi, t}(G_1, G_2)$ such that for all $t$-boundaried graphs $H$ and for all $k$:

1. $G_1 \oplus H \in G$ iff $G_2 \oplus H \in G$;
2. $(G_1 \oplus H, k) \in \Pi$ iff $(G_2 \oplus H, k + \Delta_{\Pi, t}(G_1, G_2)) \in \Pi$.

Problem $\Pi$ has FII in the class $G$ if for every integer $t$, the equivalence relation $\equiv_{\Pi, t}$ has a finite number of equivalence classes.

Main idea

If a parameterized problem has FII then its instances can be reduced by replacing any “large” protrusion by a “small” gadget (representative in a set $R_t$) from the same equivalence class.

The protrusion limit of $\Pi$ is a function $\rho_{\Pi}: \mathbb{N} \to \mathbb{N}$ defined as $\rho_{\Pi}(t) = \max_{G \in R_t} |V(G)|$.

We also define $\rho'_{\Pi}(t) = \rho_{\Pi}(2t)$. 

[Boedlaender, de Fluiter '01]
Let $\Pi$ be a parameterized graph problem restricted to a graph class $\mathcal{G}$ and let $G_1, G_2$ be two $t$-boundaried graphs in $\mathcal{G}_t$. We say that $G_1 \equiv_{\Pi, t} G_2$ if there exists a constant $\Delta_{\Pi, t}(G_1, G_2)$ such that for all $t$-boundaried graphs $H$ and for all $k$:

1. $G_1 \oplus H \in \mathcal{G}$ if and only if $G_2 \oplus H \in \mathcal{G}$;
2. $(G_1 \oplus H, k) \in \Pi$ if and only if $(G_2 \oplus H, k + \Delta_{\Pi, t}(G_1, G_2)) \in \Pi$.

Problem $\Pi$ has FII in the class $\mathcal{G}$ if for every integer $t$, the equivalence relation $\equiv_{\Pi, t}$ has a finite number of equivalence classes. The protrusion limit of $\Pi$ is a function $\rho_{\Pi}: \mathbb{N} \rightarrow \mathbb{N}$ defined as $\rho_{\Pi}(t) = \max_{G \in R_t} |V(G)|$. We also define $\rho'_{\Pi}(t) = \rho_{\Pi}(2t)$. 

[Bodlaender, de Fluiter '01]
Finite Integer Index (FII)

Let $\Pi$ be a parameterized graph problem restricted to a graph class $\mathcal{G}$ and let $G_1, G_2$ be two $t$-boundaried graphs in $\mathcal{G}_t$.

We say that $G_1 \equiv_{\Pi,t} G_2$ if there exists a constant $\Delta_{\Pi,t}(G_1, G_2)$ such that for all $t$-boundaried graphs $H$ and for all $k$:

1. $G_1 \oplus H \in \mathcal{G}$ iff $G_2 \oplus H \in \mathcal{G}$;
2. $(G_1 \oplus H, k) \in \Pi$ iff $(G_2 \oplus H, k + \Delta_{\Pi,t}(G_1, G_2)) \in \Pi$.

Problem $\Pi$ has FII in the class $\mathcal{G}$ if for every integer $t$, the equivalence relation $\equiv_{\Pi,t}$ has a finite number of equivalence classes.

Main idea

If a parameterized problem has FII then its instances can be reduced by replacing any "large" protrusion by a "small" gadget (representative in a set $R_t$) from the same equivalence class.

The protrusion limit of $\Pi$ is a function $\rho_{\Pi}: \mathbb{N} \to \mathbb{N}$ defined as $\rho_{\Pi}(t) = \max_{G \in R_t} |V(G)|$.

We also define $\rho'_{\Pi}(t) = \rho_{\Pi}(2t)$. 
Let $\Pi$ be a parameterized graph problem restricted to a graph class $\mathcal{G}$ and let $G_1, G_2$ be two $t$-boundaried graphs in $\mathcal{G}_t$.

We say that $G_1 \equiv_{\Pi,t} G_2$ if there exists a constant $\Delta_{\Pi,t}(G_1, G_2)$ such that for all $t$-boundaried graphs $H$ and for all $k$:

1. $G_1 \oplus H \in \mathcal{G}$ iff $G_2 \oplus H \in \mathcal{G}$;
2. $(G_1 \oplus H, k) \in \Pi$ iff $(G_2 \oplus H, k + \Delta_{\Pi,t}(G_1, G_2)) \in \Pi$.

Problem $\Pi$ has FII in the class $\mathcal{G}$ if for every integer $t$, the equivalence relation $\equiv_{\Pi,t}$ has a finite number of equivalence classes.
Let $\Pi$ be a parameterized graph problem restricted to a graph class $\mathcal{G}$ and let $G_1, G_2$ be two $t$-boundaried graphs in $\mathcal{G}_t$.

We say that $G_1 \equiv_{\Pi,t} G_2$ if there exists a constant $\Delta_{\Pi,t}(G_1, G_2)$ such that for all $t$-boundaried graphs $H$ and for all $k$:

1. $G_1 \oplus H \in \mathcal{G}$ iff $G_2 \oplus H \in \mathcal{G}$;
2. $(G_1 \oplus H, k) \in \Pi$ iff $(G_2 \oplus H, k + \Delta_{\Pi,t}(G_1, G_2)) \in \Pi$.

Problem $\Pi$ has FII in the class $\mathcal{G}$ if for every integer $t$, the equivalence relation $\equiv_{\Pi,t}$ has a finite number of equivalence classes.

Main idea: If a parameterized problem has FII then its instances can be reduced by replacing any “large” protrusion by a “small” gadget (representative in a set $\mathcal{R}_t$) from the same equivalence class.
Let $\Pi$ be a parameterized graph problem restricted to a graph class $\mathcal{G}$ and let $G_1, G_2$ be two $t$-boundaried graphs in $\mathcal{G}_t$.

We say that $G_1 \equiv_{\Pi,t} G_2$ if there exists a constant $\Delta_{\Pi,t}(G_1, G_2)$ such that for all $t$-boundaried graphs $H$ and for all $k$:

1. $G_1 \oplus H \in \mathcal{G}$ iff $G_2 \oplus H \in \mathcal{G}$;
2. $(G_1 \oplus H, k) \in \Pi$ iff $(G_2 \oplus H, k + \Delta_{\Pi,t}(G_1, G_2)) \in \Pi$.

Problem $\Pi$ has FII in the class $\mathcal{G}$ if for every integer $t$, the equivalence relation $\equiv_{\Pi,t}$ has a finite number of equivalence classes.

**Main idea** If a parameterized problem has FII then its instances can be reduced by replacing any “large” protrusion by a “small” gadget (representative in a set $\mathcal{R}_t$) from the same equivalence class.

The protrusion limit of $\Pi$ is a function $\rho_{\Pi}: \mathbb{N} \rightarrow \mathbb{N}$ defined as $\rho_{\Pi}(t) = \max_{G \in \mathcal{R}_t} |V(G)|$. 

[Bodlaender, de Fluiter '01]
Finite Integer Index (FII)

Let \( \Pi \) be a parameterized graph problem restricted to a graph class \( \mathcal{G} \) and let \( G_1, G_2 \) be two \( t \)-boundaried graphs in \( \mathcal{G}_t \).

We say that \( G_1 \equiv_{\Pi,t} G_2 \) if there exists a constant \( \Delta_{\Pi,t}(G_1, G_2) \) such that for all \( t \)-boundaried graphs \( H \) and for all \( k \):

1. \( G_1 \oplus H \in \mathcal{G} \) iff \( G_2 \oplus H \in \mathcal{G} \);
2. \( (G_1 \oplus H, k) \in \Pi \) iff \( (G_2 \oplus H, k + \Delta_{\Pi,t}(G_1, G_2)) \in \Pi \).

Problem \( \Pi \) has FII in the class \( \mathcal{G} \) if for every integer \( t \), the equivalence relation \( \equiv_{\Pi,t} \) has a finite number of equivalence classes.

**Main idea** If a parameterized problem has FII then its instances can be reduced by replacing any “large” protrusion by a “small” gadget (representative in a set \( \mathcal{R}_t \)) from the same equivalence class.

The protrusion limit of \( \Pi \) is a function \( \rho_{\Pi} : \mathbb{N} \to \mathbb{N} \) defined as \( \rho_{\Pi}(t) = \max_{G \in \mathcal{R}_t} |V(G)| \). We also define \( \rho'_{\Pi}(t) = \rho_{\Pi}(2t) \).
Disconnected **Planar-\mathcal{F}-Deletion** has not FII

- **We prove:** if \( \mathcal{F} \) is a family of graphs containing some disconnected graph \( H \), then **Planar-\mathcal{F}-Deletion** has not FII (in general).
Disconnected \textbf{Planar-}$\mathcal{F}$-\textbf{Deletion} \textbf{has not FII}

Let $o-\Pi$ be the non-parameterized version of \textbf{Planar-}$\mathcal{F}$-\textbf{Deletion}. Let $G_1$ and $G_2$ be two $t$-boundaried graphs.
Disconnected **PLANAR-\mathcal{F}-DELETION** has not FII

Let \( o-\Pi \) be the non-parameterized version of **PLANAR-\mathcal{F}-DELETION**. Let \( G_1 \) and \( G_2 \) be two \( t \)-boundaried graphs. We define \( G_1 \sim_{\Pi,t} G_2 \) iff

\[
\exists \text{ integer } i \text{ such that } \forall \ t\text{-boundaried graph } H, \text{ it holds }
\]

\[
\pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i,
\]

where \( \pi(G) \) denotes the **optimal value** of problem \( o-\Pi \) on graph \( G \).
Disconnected $\text{PLANAR-}\mathcal{F}\text{-DELETION}$ has not FII

- Let $o\text{-}\Pi$ be the non-parameterized version of $\text{PLANAR-}\mathcal{F}\text{-DELETION}$. Let $G_1$ and $G_2$ be two $t$-boundaried graphs. We define $G_1 \sim_{\Pi,t} G_2$ iff there exists an integer $i$ such that for all $t$-boundaried graph $H$, it holds
  \[ \pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i, \]
  where $\pi(G)$ denotes the optimal value of problem $o\text{-}\Pi$ on graph $G$.
- We let $F_1 = K_4$, $F_2 = K_{2,3}$, $F := F_1 \cup F_2$, and $\mathcal{F} = \{F\}$. 
Disconnected \textsc{Planar-}$\mathcal{F}$-\textsc{Deletion} has not FII

- Let $o$-$\Pi$ be the non-parameterized version of \textsc{Planar-}$\mathcal{F}$-\textsc{Deletion}. Let $G_1$ and $G_2$ be two $t$-boundaried graphs. We define $G_1 \sim_{\Pi, t} G_2$ iff there exists an integer $i$ such that for every $t$-boundaried graph $H$, it holds
  \[ \pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i, \]
  where $\pi(G)$ denotes the optimal value of problem $o$-$\Pi$ on graph $G$.
- We let $F_1 = K_4$, $F_2 = K_{2,3}$, $F := F_1 \uplus F_2$, and $\mathcal{F} = \{F\}$.
- For $i \geq 1$, let $G_i$ (resp. $H_i$) be the 1-boundaried graph consisting of a boundary vertex $v$ (resp. $u$) together with $i$ disjoint copies of $F_1$ (resp. $F_2$) joined to $v$ (resp. $u$) by an edge.
Disconnected **Planar-\(F\)-Deletion** has not FII

- Let \(o-\Pi\) be the non-parameterized version of **Planar-\(F\)-Deletion**. Let \(G_1\) and \(G_2\) be two \(t\)-boundaried graphs. We define \(G_1 \sim_{\Pi,t} G_2\) iff \(\exists\) integer \(i\) such that \(\forall\ t\)-boundaried graph \(H\), it holds

\[
\pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i,
\]

where \(\pi(G)\) denotes the *optimal value* of problem \(o-\Pi\) on graph \(G\).

- We let \(F_1 = K_4\), \(F_2 = K_{2,3}\), \(F := F_1 \uplus F_2\), and \(\mathcal{F} = \{F\}\).

- For \(i \geq 1\), let \(G_i\) (resp. \(H_i\)) be the 1-boundaried graph consisting of a boundary vertex \(v\) (resp. \(u\)) together with \(i\) disjoint copies of \(F_1\) (resp. \(F_2\)) joined to \(v\) (resp. \(u\)) by an edge.

- By construction, if \(i, j \geq 1\), it holds \(\pi(G_i \oplus H_j) = \min\{i, j\}\).
Disconnected \textbf{Planar-}$\mathcal{F}$-\textbf{Deletion} has not FII

- Let \( o-\Pi \) be the non-parameterized version of \textbf{Planar-}$\mathcal{F}$-\textbf{Deletion}.
  Let \( G_1 \) and \( G_2 \) be two \( t \)-boundaried graphs. We define \( G_1 \sim_{\Pi,t} G_2 \) iff
  \( \exists \) integer \( i \) such that \( \forall \) \( t \)-boundaried graph \( H \), it holds
  \[
  \pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i,
  \]
  where \( \pi(G) \) denotes the \textbf{optimal value} of problem \( o-\Pi \) on graph \( G \).

- We let \( F_1 = K_4, F_2 = K_{2,3} \), \( F := F_1 \cup F_2 \), and \( \mathcal{F} = \{ F \} \).

- For \( i \geq 1 \), let \( G_i \) (resp. \( H_i \)) be the \( 1 \)-boundaried graph consisting of a
  boundary vertex \( v \) (resp. \( u \)) together with \( i \) disjoint copies of \( F_1 \)
  (resp. \( F_2 \)) joined to \( v \) (resp. \( u \)) by an edge.

- By construction, if \( i, j \geq 1 \), it holds \( \pi(G_i \oplus H_j) = \min\{i,j\} \).

- Then, if we take \( 1 \leq n < m \),
  \[
  \pi(G_n \oplus H_{n-1}) - \pi(G_m \oplus H_{n-1}) = (n - 1) - (n - 1) = 0,
  \]
  \[
  \pi(G_n \oplus H_m) - \pi(G_m \oplus H_m) = n - m < 0.
  \]
Disconnected \textbf{Planar-\(\mathcal{F}\)-Deletion} has not FII

- Let \(o-\Pi\) be the non-parameterized version of \textbf{Planar-\(\mathcal{F}\)-Deletion}.
- Let \(G_1\) and \(G_2\) be two \(t\)-boundaried graphs. We define \(G_1 \sim_{\Pi,t} G_2\) iff \(\exists\) integer \(i\) such that \(\forall\) \(t\)-boundaried graph \(H\), it holds
  \[
  \pi(G_1 \oplus H) = \pi(G_2 \oplus H) + i,
  \]
  where \(\pi(G)\) denotes the optimal value of problem \(o-\Pi\) on graph \(G\).

- We let \(F_1 = K_4\), \(F_2 = K_{2,3}\), \(F := F_1 \cup F_2\), and \(\mathcal{F} = \{F\}\).

- For \(i \geq 1\), let \(G_i\) (resp. \(H_i\)) be the \(1\)-boundaried graph consisting of a boundary vertex \(v\) (resp. \(u\)) together with \(i\) disjoint copies of \(F_1\) (resp. \(F_2\)) joined to \(v\) (resp. \(u\)) by an edge.

- By construction, if \(i, j \geq 1\), it holds \(\pi(G_i \oplus H_j) = \min\{i, j\}\).

- Then, if we take \(1 \leq n < m\),
  \[
  \pi(G_n \oplus H_{n-1}) - \pi(G_m \oplus H_{n-1}) = (n - 1) - (n - 1) = 0,
  \]
  \[
  \pi(G_n \oplus H_m) - \pi(G_m \oplus H_m) = n - m < 0.
  \]

- Thus, \(G_n, G_m \notin\) same equiv. class of \(\sim_{\Pi,1}\) whenever \(1 \leq n \leq m\).
Some important ingredients  (suppose problem $\Pi$ has FII)

Lemma (The parameter does not increase)

$\forall$ fixed $t$, $\exists$ finite set $R_t$ of $t$-boundaried graphs s.t. for each $t$-boundaried graph $G \in G_t$ $\exists G' \in R_t$ s.t. $G \equiv \Pi, t$, $G'$ and $\Delta \Pi, t (G, G') \geq 0$.

Lemma (Finding maximum sized protrusions)

Let $t$ be a constant. Given an $n$-vertex graph $G$, a $t$-protrusion of $G$ with the maximum number of vertices can be found in time $O(n^t + 1)$.

Lemma (Big... but not too big!)

If one is given a $t$-protrusion $X \subseteq V(G)$ s.t. $\rho' \Pi(t) < |X|$, then one can, in time $O(|X|)$, find an equiv. $2^t$-protrusion $W$ s.t. $\rho' \Pi(t) < |W| \leq 2^t \cdot \rho' \Pi(t)$.

Lemma (Replacing protrusions of constant size)

For $t \in \mathbb{N}$, suppose that the set $R_t$ of representatives of $\equiv \Pi, t$ is given. If $W$ is a $t$-protrusion of size at most a fixed constant $c$, then one can decide in constant time which $G' \in R_t$ satisfies $G' \equiv \Pi, t [W]$. 

44/50
Lemma (The parameter does not increase)

∀ fixed t, ∃ finite set \( \mathcal{R}_t \) of t-boundaried graphs s.t. for each t-boundaried graph \( G \in \mathcal{G}_t \) ∃ \( G' \in \mathcal{R}_t \) s.t. \( G \equiv_{\Pi, t} G' \) and \( \Delta_{\Pi, t}(G, G') \geq 0 \).
Some important ingredients  (suppose problem $\Pi$ has FII)

**Lemma (The parameter does not increase)**

$\forall$ fixed $t$, $\exists$ finite set $R_t$ of $t$-boundaried graphs s.t. for each $t$-boundaried graph $G \in G_t$ $\exists G' \in R_t$ s.t. $G \equiv_{\Pi,t} G'$ and $\Delta_{\Pi,t}(G, G') \geq 0$.

**Lemma (Finding maximum sized protrusions)**

Let $t$ be a constant. Given an $n$-vertex graph $G$, a $t$-protrusion of $G$ with the maximum number of vertices can be found in time $O(n^{t+1})$.
Some important ingredients  
(suppose problem \( \Pi \) has FII)

**Lemma (The parameter does not increase)**
\[ \forall \text{ fixed } t, \exists \text{ finite set } R_t \text{ of } t\text{-boundaried graphs s.t. for each } t\text{-boundaried graph } G \in \mathcal{G}_t \exists G' \in R_t \text{ s.t. } G \equiv_{\Pi, t} G' \text{ and } \Delta_{\Pi, t}(G, G') \geq 0. \]

**Lemma (Finding maximum sized protrusions)**
Let \( t \) be a constant. Given an \( n \)-vertex graph \( G \), a \( t \)-protrusion of \( G \) with the maximum number of vertices can be found in time \( O(n^{t+1}) \).

**Lemma (Big... but not too big!)**
If one is given a \( t \)-protrusion \( X \subseteq V(G) \) s.t. \( \rho'_\Pi(t) < |X| \), then one can, in time \( O(|X|) \), find an equiv. \( 2t \)-protrusion \( W \) s.t. \( \rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t) \).
Lemma (The parameter does not increase)

∀ fixed t, ∃ finite set \( R_t \) of t-boundaried graphs s.t. for each t-boundaried graph \( G \in G_t \) \( \exists G' \in R_t \) s.t. \( G \equiv_{\Pi, t} G' \) and \( \Delta_{\Pi, t}(G, G') \geq 0 \).

Lemma (Finding maximum sized protrusions)

Let \( t \) be a constant. Given an n-vertex graph \( G \), a \( t \)-protrusion of \( G \) with the maximum number of vertices can be found in time \( O(n^{t+1}) \).

Lemma (Big... but not too big!)

If one is given a \( t \)-protrusion \( X \subseteq V(G) \) s.t. \( \rho'_\Pi(t) < |X| \), then one can, in time \( O(|X|) \), find an equiv. \( 2t \)-protrusion \( W \) s.t. \( \rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t) \).

Lemma (Replacing protrusions of constant size)

For \( t \in \mathbb{N} \), suppose that the set \( R_t \) of representatives of \( \equiv_{\Pi, t} \) is given. If \( W \) is a \( t \)-protrusion of size at most a fixed constant \( c \), then one can decide in constant time which \( G' \in R_t \) satisfies \( G' \equiv_{\Pi, t} G[W] \).
Protrusion replacement

Protrusion reduction rule

- Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).

Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho'_{\Pi(t)}\).

Let \(W \subseteq V(G)\) be a \(2t\)-protrusion of \(G\) s.t. \(\rho'_{\Pi(t)} < |W| \leq 2 \cdot \rho'_{\Pi(t)}\).

We let \(G_W\) denote the \(2t\)-boundaried graph \(G[W]\) with boundary \(bd(G_W) = \partial G(W)\).

Let further \(G_1 \in R_{2t}\) be the representative of \(G_W\) for the equivalence relation \(\equiv_{\Pi(t)}\), |

The protrusion reduction rule (for boundary size \(t\)) is the following:

Reduce \((G, k)\) to \((G', k') = (G[W] \oplus G_1, k - \Delta_{\Pi(t)}, 2t(G_1, GW))\).

It runs in polynomial time... given the sets of representatives!
Protrusion replacement

**Protrusion reduction rule**

- Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).
- Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho'_\Pi(t)|.\)
Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).

Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho'_\Pi(t)\).

Let \(W \subseteq V(G)\) be a \(2t\)-protrusion of \(G\) s.t. \(\rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t)\).
Protrusion replacement

**Protrusion reduction rule**

- Let \( (G, k) \in \Pi \) and let \( t \in \mathbb{N} \) be a constant (to be fixed later).
- Suppose that \( G \) has a \( t \)-protrusion \( W' \subseteq V(G) \) s.t. \( |W'| > \rho'_\Pi(t) \).
- Let \( W \subseteq V(G) \) be a \( 2t \)-protrusion of \( G \) s.t. \( \rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t) \).
- We let \( G_W \) denote the \( 2t \)-boundaried graph \( G[W] \) with boundary \( \text{bd}(G_W) = \partial_G(W) \).
Protrusion replacement

**Protrusion reduction rule**

- Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).
- Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho_\Pi'(t)\).
- Let \(W \subseteq V(G)\) be a \(2t\)-protrusion of \(G\) s.t. \(\rho_\Pi'(t) < |W| \leq 2 \cdot \rho_\Pi'(t)\).
- We let \(G_W\) denote the \(2t\)-boundaried graph \(G[W]\) with boundary \(bd(G_W) = \partial_G(W)\).
- Let further \(G_1 \in R_{2t}\) be the representative of \(G_W\) for the equivalence relation \(\equiv_{\Pi, |\partial(W)|}\).
Protrusion replacement

**Protrusion reduction rule**

- Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).
- Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho'_\Pi(t)\).
- Let \(W \subseteq V(G)\) be a \(2t\)-protrusion of \(G\) s.t. \(\rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t)\).
- We let \(G_W\) denote the \(2t\)-boundaried graph \(G[W]\) with boundary \(\text{bd}(G_W) = \partial_G(W)\).
- Let further \(G_1 \in \mathcal{R}_{2t}\) be the representative of \(G_W\) for the equivalence relation \(\equiv_{\Pi, |\partial(W)|}\).
- The protrustion reduction rule (for boundary size \(t\)) is the following:
  
  Reduce \((G, k)\) to \((G', k') = (G[V \setminus W] \oplus G_1, k - \Delta_{\Pi,2t}(G_1, G_W))\).
Protrusion replacement

**Protrusion reduction rule**

- Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).
- Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho'_\Pi(t)\).
- Let \(W \subseteq V(G)\) be a \(2t\)-protrusion of \(G\) s.t. \(\rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t)\).
- We let \(G_W\) denote the \(2t\)-boundaried graph \(G[W]\) with boundary \(\text{bd}(G_W) = \partial G(W)\).
- Let further \(G_1 \in \mathcal{R}_{2t}\) be the representative of \(G_W\) for the equivalence relation \(\equiv_{\Pi, |\partial(W)|}\).
- The protrusion reduction rule (for boundary size \(t\)) is the following:
  \[
  \text{Reduce } (G, k) \text{ to } (G', k') = (G[V \setminus W] \oplus G_1, k - \Delta_{\Pi,2t}(G_1, G_W)).
  \]

It runs in polynomial time...
Protrusion replacement

### Protrusion reduction rule

- Let \((G, k) \in \Pi\) and let \(t \in \mathbb{N}\) be a constant (to be fixed later).
- Suppose that \(G\) has a \(t\)-protrusion \(W' \subseteq V(G)\) s.t. \(|W'| > \rho'_\Pi(t)\).
- Let \(W \subseteq V(G)\) be a \(2t\)-protrusion of \(G\) s.t. \(\rho'_\Pi(t) < |W| \leq 2 \cdot \rho'_\Pi(t)\).
- We let \(G_W\) denote the \(2t\)-boundaried graph \(G[W]\) with boundary \(\text{bd}(G_W) = \partial_G(W)\).
- Let further \(G_1 \in \mathcal{R}_{2t}\) be the representative of \(G_W\) for the equivalence relation \(\equiv_{\Pi, |\partial(W)|}\).
- The protrusion reduction rule (for boundary size \(t\)) is the following:

\[
\text{Reduce } (G, k) \text{ to } (G', k') = (G[V \setminus W] \oplus G_1, k - \Delta_{\Pi, 2t}(G_1, G_W)).
\]

It runs in \textbf{polynomial time}... given the sets of representatives!
An \((\alpha, t)\)-protrusion decomposition of a graph \(G\) is a partition \(\mathcal{P} = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell\) of \(V(G)\) such that:

- for every \(1 \leq i \leq \ell\), \(N(Y_i) \subseteq Y_0\);
- for every \(1 \leq i \leq \ell\), \(Y_i \cup N_{Y_0}(Y_i)\) is a \(t\)-protrusion of \(G\);
- \(\max\{\ell, |Y_0|\} \leq \alpha\).
We apply exhaustively the protrusion replacement rule.
We apply exhaustively the protrusion replacement rule.

If \((G, k)\) is reduced w.r.t. the protrusion reduction rule with boundary size \(\beta\) (this can be done in polynomial time), \(\forall t \leq \beta\), every \(t\)-protrusion \(W\) of \(G\) has size \(\leq \rho'_\Pi(t)\).
We apply exhaustively the protrusion replacement rule.

If \((G, k)\) is reduced w.r.t. the protrusion reduction rule with boundary size \(\beta\) (this can be done in polynomial time), \(\forall t \leq \beta\), every \(t\)-protrusion \(W\) of \(G\) has size \(\leq \rho'_\Pi(t)\).

We can choose \(\beta := 2t + \omega(H)\), where \(t\) comes from the treewidth-bounding property of \(\Pi\).
1. We apply **exhaustively** the **protrusion replacement rule**.

If \((G, k)\) is **reduced** w.r.t. the protrusion reduction rule with boundary size \(\beta\) (this can be done in **polynomial time**), \(\forall t \leq \beta\), every \(t\)-protrusion \(W\) of \(G\) has size \(\leq \rho'_\Pi(t)\).

We can choose \(\beta := 2t + \omega(H)\), where \(t\) comes from the **treewidth-bounding** property of \(\Pi\).

2. We use **protrusion decompositions** to analyze the kernel size.
Kernelization algorithm

1. We apply exhaustively the protrusion replacement rule.

   If \((G, k)\) is reduced w.r.t. the protrusion reduction rule with boundary size \(\beta\) (this can be done in polynomial time), \(\forall t \leq \beta\), every \(t\)-protrusion \(W\) of \(G\) has size \(\leq \rho'_{\Pi}(t)\).

   We can choose \(\beta := 2t + \omega(H)\), where \(t\) comes from the treewidth-bounding property of \(\Pi\).

2. We use protrusion decompositions to analyze the kernel size.

   Using what we explained before, we can easily prove that:

   Let \(\Pi\) be a parameterized graph problem that has FII and is \(t\)-treewidth-bounding, both on the class of \(H\)-topological-minor-free graphs.
Kernelization algorithm

1. We apply exhaustively the protrusion replacement rule.

If \((G, k)\) is reduced w.r.t. the protrusion reduction rule with boundary size \(\beta\) (this can be done in polynomial time), \(\forall t \leq \beta\), every \(t\)-protrusion \(W\) of \(G\) has size \(\leq \rho'_\Pi(t)\).

We can choose \(\beta := 2t + \omega(H)\), where \(t\) comes from the treewidth-bounding property of \(\Pi\).

2. We use protrusion decompositions to analyze the kernel size.

Using what we explained before, we can easily prove that:

Let \(\Pi\) be a parameterized graph problem that has FII and is \(t\)-treewidth-bounding, both on the class of \(H\)-topological-minor-free graphs. Then any reduced \(\text{YES}\)-instance \((G, k)\) has a protrusion decomposition \(V(G) = Y_0 \cup Y_1 \cup \cdots \cup Y_\ell\) s.t.:

1. \(|Y_0| = O(k)|; 
2. \(|Y_i| \leq \rho'_\Pi(2t + \omega_H)\) for \(1 \leq i \leq \ell\); and 
3. \(\ell = O(k)|.

1 Preliminaries

2 Protrusion decompositions
   • Definitions
   • A simple algorithm to compute them

3 Single-exponential algorithm for Planar-$\mathcal{F}$-Deletion
   • Motivation and our result
   • Sketch of proof
   • Further research

4 Linear kernels on graphs without topological minors
   • Motivation and our result
   • Idea of proof
   • Further research
Limits of our approach and further research

- For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?
Limits of our approach and further research

- For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

1. A class $\mathcal{G}$ of graphs locally excludes a minor if $\forall r \in \mathbb{N}, \exists H_r$ s.t. the $r$-neighborhood of a vertex of any graph of $\mathcal{G}$ excludes $H_r$ as a minor.

   (includes $H$-minor-free but incomparable with $H$-topological-minor-free)
Limits of our approach and further research

- For which notions of sparseness (beyond \(H\)-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

1. A class \(G\) of graphs **locally excludes a minor** if \(\forall r \in \mathbb{N}, \exists H_r\) s.t. the \(r\)-neighborhood of a vertex of any graph of \(G\) excludes \(H_r\) as a minor.

(includes \(H\)-minor-free but incomparable with \(H\)-topological-minor-free)

Except for a very restricted case, our technique **fails**.
Limits of our approach and further research

- For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

1. A class $\mathcal{G}$ of graphs **locally excludes a minor** if $\forall r \in \mathbb{N}, \exists H_r$ s.t. the $r$-neighborhood of a vertex of any graph of $\mathcal{G}$ excludes $H_r$ as a minor.

   (includes $H$-minor-free but incomparable with $H$-topological-minor-free)

   Except for a very restricted case, our technique fails.

2. Graphs of **bounded expansion** (contains $H$-topological-minor-free)?

Obtaining a kernel for Treewidth-$t$ Vertex Deletion on graphs of bounded expansion is as hard as on general graphs. Best known kernel: $kO(t)$. [Fomin, Lokshtanov, Misra, Saurabh '12] Constructing the kernels? Finding the sets of representatives!! Explicit constants? Lower bounds on their size? 49/50
For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

1. A class $\mathcal{G}$ of graphs \textbf{locally excludes a minor} if $\forall r \in \mathbb{N}, \exists H_r$ s.t. the $r$-neighborhood of a vertex of any graph of $\mathcal{G}$ excludes $H_r$ as a minor. (includes $H$-minor-free but incomparable with $H$-topological-minor-free)

Except for a very \textbf{restricted} case, our technique \textbf{fails}.

2. Graphs of \textbf{bounded expansion} (contains $H$-topological-minor-free)?

Obtaining a kernel for \textsc{Treewidth-$t$ Vertex Deletion} on graphs of \textbf{bounded expansion} is \textbf{as hard} as on \textbf{general graphs}.
Limits of our approach and further research

- For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

  1. A class $G$ of graphs **locally excludes a minor** if $\forall r \in \mathbb{N}, \exists H_r$ s.t. the $r$-neighborhood of a vertex of any graph of $G$ excludes $H_r$ as a minor. (includes $H$-minor-free but incomparable with $H$-topological-minor-free)

    Except for a very restricted case, our technique **fails**.

  2. Graphs of **bounded expansion** (contains $H$-topological-minor-free)?

    Obtaining a kernel for TreeWidth-$t$ Vertex Deletion on graphs of bounded expansion is as hard as on general graphs.

    Best known kernel: $k^{O(t)}$. [Fomin, Lokshtanov, Misra, Saurabh '12]
Limits of our approach and further research

For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

1. A class $G$ of graphs **locally excludes a minor** if $\forall r \in \mathbb{N}, \exists H_r$ s.t. the $r$-neighborhood of a vertex of any graph of $G$ excludes $H_r$ as a minor.

   (includes $H$-minor-free but incomparable with $H$-topological-minor-free)

   Except for a very **restricted** case, our technique **fails**.

2. Graphs of **bounded expansion** (contains $H$-topological-minor-free)?

   Obtaining a kernel for TreeWidth-$t$ Vertex Deletion on graphs of bounded expansion is as hard as on general graphs.

   Best known kernel: $k^{O(t)}$.  

   [Fomin, Lokshtanov, Misra, Saurabh '12]

- **Constructing** the kernels? **Finding the sets of representatives!!**
Limits of our approach and further research

- For which notions of sparseness (beyond $H$-topological-minor-free graphs) can we use our technique to obtain polynomial kernels?

1. A class $\mathcal{G}$ of graphs **locally excludes a minor** if $\forall r \in \mathbb{N}, \exists H_r$ s.t. the $r$-neighborhood of a vertex of any graph of $\mathcal{G}$ excludes $H_r$ as a minor. (includes $H$-minor-free but incomparable with $H$-topological-minor-free)

Except for a very restricted case, our technique **fails**.

2. Graphs of **bounded expansion** (contains $H$-topological-minor-free)?

Obtaining a kernel for $\text{Treewidth}-t \ 	ext{Vertex Deletion}$ on graphs of bounded expansion is as hard as on general graphs.

Best known kernel: $k^{O(t)}$. [Fomin, Lokshtanov, Misra, Saurabh ’12]

- Constructing the kernels? Finding the sets of representatives!!

- Explicit constants? Lower bounds on their size?
Gràcies!