Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism

(but were afraid to ask)

Dániel Marx, Michał Pilipczuk

STACS'14, Lyon,
March 6$^{\text{th}}$, 2014

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

- NP-hard, as it generalizes Hamiltonian Path.

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

- NP-hard, as it generalizes Hamiltonian Path.
- **Natural question**: What if H is 'small', or 'simple'.

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

- NP-hard, as it generalizes Hamiltonian Path.
- **Natural question**: What if H is 'small', or 'simple'.
- **Parameterized complexity**

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

- NP-hard, as it generalizes Hamiltonian Path.
- **Natural question**: What if H is 'small', or 'simple'.
- **Parameterized complexity**
  - Let $k := |V(H)|$.

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

- NP-hard, as it generalizes Hamiltonian Path.
- **Natural question**: What if H is 'small', or 'simple'.
- **Parameterized complexity**
  - Let $k := |V(H)|$.
  - Trivial $n^{\mathcal{O}(k)}$ algorithm.

The problem

Subgraph Isomorphism

Given H and G, decide if H is a subgraph of G.

- NP-hard, as it generalizes Hamiltonian Path.
- **Natural question**: What if H is 'small', or 'simple'.
- **Parameterized complexity**
  - Let $k := |V(H)|$.
  - Trivial $n^{\mathcal{O}(k)}$ algorithm.
  - Generalization of Clique $\Rightarrow$ no $f(k) \cdot n^{\mathcal{O}(1)}$ algorithm (FPT), unless FPT=W[1].

Tree-like H

- But if H is a **path** on k vertices, then there is a $1.66^k \cdot n^{\mathcal{O}(1)}$ algorithm [Björklund].

Tree-like H

- But if H is a **path** on k vertices, then there is a $1.66^k \cdot n^{\mathcal{O}(1)}$ algorithm [Björklund].
- **Idea**: measure the *treewidth* of H (denoted $\mathrm{tw}(H)$).

Tree-like H

- But if H is a **path** on k vertices, then there is a $1.66^k \cdot n^{\mathcal{O}(1)}$ algorithm [Björklund].
- **Idea**: measure the *treewidth* of H (denoted $\mathrm{tw}(H)$).

Theorem [Alon, Yuster, Zwick]

Subgraph Isomorphism is solvable in time $2^{\mathcal{O}(|V(H)|)} \cdot n^{\mathcal{O}(\mathrm{tw}(H))}$.

Tree-like H

- But if H is a **path** on k vertices, then there is a $1.66^k \cdot n^{\mathcal{O}(1)}$ algorithm [Björklund].
- **Idea**: measure the *treewidth* of H (denoted $\mathrm{tw}(H)$).

Theorem [Alon, Yuster, Zwick]

Subgraph Isomorphism is solvable in time $2^{\mathcal{O}(|V(H)|)} \cdot n^{\mathcal{O}(\mathrm{tw}(H))}$.

- We need **small** $|V(H)|$ and **even smaller** $\mathrm{tw}(H)$, but G can be arbitrarily complicated.

Tree-like H

- But if H is a **path** on k vertices, then there is a $1.66^k \cdot n^{\mathcal{O}(1)}$ algorithm [Björklund].
- **Idea**: measure the *treewidth* of H (denoted $\mathrm{tw}(H)$).

Theorem [Alon, Yuster, Zwick]

Subgraph Isomorphism is solvable in time $2^{\mathcal{O}(|V(H)|)} \cdot n^{\mathcal{O}(\mathrm{tw}(H))}$.

- We need **small** $|V(H)|$ and **even smaller** $\mathrm{tw}(H)$, but G can be arbitrarily complicated.
- **Question**: Could we get something better if we assume that $\mathrm{tw}(G)$ is small, instead of $\mathrm{tw}(H)$?

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

- We need small $\mathrm{tw}(G)$, but $|V(H)|$ must be also small.

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

- We need small $\mathrm{tw}(G)$, but $|V(H)|$ must be also small.

Theorem [Matoušek, Thomas]

Subgraph Isomorphism for connected H can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for some computable function f.

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

- We need small $\mathrm{tw}(G)$, but $|V(H)|$ must be also small.

Theorem [Matoušek, Thomas]

Subgraph Isomorphism for connected H can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for some computable function f.

- We can have unbounded size of $V(H)$, but at the cost of:

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

- We need small $\mathrm{tw}(G)$, but $|V(H)|$ must be also small.

Theorem [Matoušek, Thomas]

Subgraph Isomorphism for connected H can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for some computable function f.

- We can have unbounded size of $V(H)$, but at the cost of:
  - bounding $\Delta(H)$;

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

- We need small $\mathrm{tw}(G)$, but $|V(H)|$ must be also small.

Theorem [Matoušek, Thomas]

Subgraph Isomorphism for connected H can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for some computable function f.

- We can have unbounded size of $V(H)$, but at the cost of:
  - bounding $\Delta(H)$;
  - having $\mathrm{tw}(G)$ in the exponent;

Tree-like G

Theorem [FO model checking]

Subgraph Isomorphism can be solved in time $f(|V(H)|, \mathrm{tw}(G)) \cdot n$ for some computable function f.

- We need small $\mathrm{tw}(G)$, but $|V(H)|$ must be also small.

Theorem [Matoušek, Thomas]

Subgraph Isomorphism for connected H can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for some computable function f.

- We can have unbounded size of $V(H)$, but at the cost of:
  - bounding $\Delta(H)$;
  - having $\mathrm{tw}(G)$ in the exponent;
  - assuming that H is connected.

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...
  - whether it's treewidth of H or of G;

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...
  - whether it's treewidth of H or of G;
  - what other assumptions about other parameters we make.

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...
  - whether it's treewidth of H or of G;
  - what other assumptions about other parameters we make.
- What about other constraints? Like planarity, cliquewidth, excluded minors...

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...
  - whether it's treewidth of H or of G;
  - what other assumptions about other parameters we make.
- What about other constraints? Like planarity, cliquewidth, excluded minors...
- **Our motivation**:

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...
  - whether it's treewidth of H or of G;
  - what other assumptions about other parameters we make.
- What about other constraints? Like planarity, cliquewidth, excluded minors...
- **Our motivation**:
  - **It's a mess.**

Motivation

- **Question**: How does treewidth influence the parameterized complexity of Subgraph Isomorphism?
- **Answer**: It depence...
  - whether it's treewidth of H or of G;
  - what other assumptions about other parameters we make.
- What about other constraints? Like planarity, cliquewidth, excluded minors...
- **Our motivation**:
  - **It's a mess.**
  - Let's try to clean it.

Parameters

We consider the following 10 parameters for H and G:

1. Number of vertices $|V(\cdot)|$ (only H).
2. Number of connected components $\mathrm{cc}(\cdot)$.
3. Maximum degree $\Delta(\cdot)$.
4. Treewidth $\mathrm{tw}(\cdot)$.
5. Pathwidth $\mathrm{pw}(\cdot)$.
6. Feedback vertex set number $\mathrm{fvs}(\cdot)$.
7. Cliquewidth $\mathrm{cw}(\cdot)$.
8. Genus $\mathrm{genus}(\cdot)$.
9. Hadwiger number (largest clique minor) $\mathrm{hadw}(\cdot)$.
10. Topological Hadwiger number (largest topological clique minor) $\mathrm{hadw}_T(\cdot)$.

Constraints

- Known results indicate that there is a complexity shift change between $\mathrm{tw}(G) = 1$ and $\mathrm{tw}(G) = 2$.

Constraints

- Known results indicate that there is a complexity shift change between $\mathrm{tw}(G) = 1$ and $\mathrm{tw}(G) = 2$.
- Also the case of **planar** graphs (genus $= 0$) be of interest.

Constraints

- Known results indicate that there is a complexity shift change between $\mathrm{tw}(G) = 1$ and $\mathrm{tw}(G) = 2$.
- Also the case of **planar** graphs (genus $= 0$) be of interest.

5 possible additional restrictions on H or G:

1. Genus is 0 (i.e., planar).
2. Number of components is 1 (i.e., connected).
3. Treewidth is at most 1 (i.e., graph is a forest).
4. Maximum degree at most 2 (i.e., paths and cycles).
5. Maximum degree at most 3.

Goal

Goal

Determine for every combination of these parameters and constraints, whether there is an algorithm with running time

$$f_1(p_1, p_2, \ldots, p_\ell) \cdot n^{f_2(p_{\ell+1}, \ldots, p_t)},$$

working under given constraints $c_1, c_2, \ldots, c_q$.

Goal

Goal

Determine for every combination of these parameters and constraints, whether there is an algorithm with running time

$$f_1(p_1, p_2, \ldots, p_\ell) \cdot n^{f_2(p_{\ell+1}, \ldots, p_t)},$$

working under given constraints $c_1, c_2, \ldots, c_q$.

- Vectors $(p_1, \ldots, p_\ell)$, $(p_{\ell+1}, \ldots, p_t)$, $(c_1, \ldots, c_q)$, are called the *description*.

Implications

- There are some easy implications between the results.

Implications

- There are some easy implications between the results.
- An $f(|V(H)|, \mathrm{cw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm implies an $f(|V(H)|, \mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm:

Implications

- There are some easy implications between the results.
- An $f(|V(H)|, \mathrm{cw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm implies an $f(|V(H)|, \mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm:
  - Replacing a parameter with a smaller one makes the description *stronger*.

Implications

- There are some easy implications between the results.
- An $f(|V(H)|, \mathrm{cw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm implies an $f(|V(H)|, \mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm:
  - Replacing a parameter with a smaller one makes the description *stronger*.
- If we assume that $\Delta(G) \leq 3$, then also $\Delta(H) \leq 3$:

Implications

- There are some easy implications between the results.
- An $f(|V(H)|, \mathrm{cw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm implies an $f(|V(H)|, \mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm:
  - Replacing a parameter with a smaller one makes the description *stronger*.
- If we assume that $\Delta(G) \leq 3$, then also $\Delta(H) \leq 3$:
  - Some parameters/constraints for G imply some parameters/constraints for H.

Implications

- There are some easy implications between the results.
- An $f(|V(H)|, \mathrm{cw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm implies an $f(|V(H)|, \mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm:
  - Replacing a parameter with a smaller one makes the description *stronger*.
- If we assume that $\Delta(G) \leq 3$, then also $\Delta(H) \leq 3$:
  - Some parameters/constraints for G imply some parameters/constraints for H.
- Thus we obtain a **partial order** of descriptions.

Implications

- There are some easy implications between the results.
- An $f(|V(H)|, \mathrm{cw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm implies an $f(|V(H)|, \mathrm{tw}(G)) \cdot n^{\mathcal{O}(1)}$ algorithm:
  - Replacing a parameter with a smaller one makes the description *stronger*.
- If we assume that $\Delta(G) \leq 3$, then also $\Delta(H) \leq 3$:
  - Some parameters/constraints for G imply some parameters/constraints for H.
- Thus we obtain a **partial order** of descriptions.
- There are $> 3^{19}$ descriptions, but maybe a smaller set of results explains all of them.

Main result

Main result

For any combination of the 19 parameters in the multiplier and the exponent and for any combinations of the 5 additional restrictions, we either show an algorithm or prove that no such algorithm exists (under P$\neq$NP or FPT$\neq$W[1]).

Main result

Main result

For any combination of the 19 parameters in the multiplier and the exponent and for any combinations of the 5 additional restrictions, we either show an algorithm or prove that no such algorithm exists (under P$\neq$NP or FPT$\neq$W[1]).

- All the cases are explained by 11 maximally-positive and 17 minimally-negative results.

Main result

Main result

For any combination of the 19 parameters in the multiplier and the exponent and for any combinations of the 5 additional restrictions, we either show an algorithm or prove that no such algorithm exists (under P$\neq$NP or FPT$\neq$W[1]).

- All the cases are explained by 11 maximally-positive and 17 minimally-negative results.
- Out of these, 4 algorithmic results are new: they are contained in 2 new families of tractable cases of Subgraph Isomorphism.

Main result

Main result

For any combination of the 19 parameters in the multiplier and the exponent and for any combinations of the 5 additional restrictions, we either show an algorithm or prove that no such algorithm exists (under P$\neq$NP or FPT$\neq$W[1]).

- All the cases are explained by 11 maximally-positive and 17 minimally-negative results.
- Out of these, 4 algorithmic results are new: they are contained in 2 new families of tractable cases of Subgraph Isomorphism.
- 11 negative results are new: a new methodology for proving intractability on planar and bounded-genus graphs.

Results

Short Description	Thm	H										G								
		$\lvert V(\cdot)\rvert$	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_T	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_T
FO model checking	Thm P.1 (page 17)	M															M			
	Thm P.2 (page 17)	M																		M
Color coding	Thm P.3 (page 17)	M					E													
Matoušek-Thomas	Thm P.4 (page 18)		M	M												E				
Paths&Cycles $\to$ Paths&Cycles	Thm P.5 (page 18)											E	2							
Dynamic Programming	Thm P.6 (page 19)		E	2												M				
	Thm P.7 (page 21)		E	2													E			
	Thm P.8* (page 23)		M													1				
FVS and CSPs	Thm P.9* (page 28)		M	2									M	M						
	Thm P.10* (page 36)		E										M	M				E		
	Thm P.11* (page 46)		E	E									M	M					E	
Bin Packing	Thm N.1 (page 46)											M	2			1				
	Thm N.2 (page 47)		1				1							E	E			0		
	Thm N.3 (page 47)			2								1	3		E	1				
Planar cubic HamPath	Thm N.4 (page 48)		1	2			1						3					0		
Clique	Thm N.5 (page 48)	M	1					E												
HamPath in bounded **cw**	Thm N.6 (page 48)		1	2			1										M			
Grid Tiling, 1-in-n gadgets	Thm N.7* (page 57)		M			E	1					1	3	M	M			0		
	Thm N.8* (page 61)		1			E	1						M	M	M			M	E	
	Thm N.9* (page 61)		1			E	1						3	M	M			M		
	Thm N.10* (page 63)		1	3		E	1						M	M	M		E	M		
Grid Tiling, moustache gadgets	Thm N.11* (page 64)		1	3		E	1							M	M			0		
	Thm N.12* (page 66)		1			E	1						3		M			0		
Small planar graph	Thm N.13* (page 67)	M	1	3					0											
Exact Planar Arc Supply	Thm N.14* (page 74)		M	2			1					1		M	M			0		
	Thm N.15* (page 77)		M	2			1					1	3		M			0		
	Thm N.16* (page 79)		M	2			1					1		M	M		E	M		
	Thm N.17* (page 79)		M	2			1					1	M		M		E	M		

Figure 1: Positive and negative results in the paper. Results marked with * are new findings that were not known before.

How did we do it?

- How do you manage such a project?

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**
  - **Problem**: Checking coverage of the descriptions is NP-hard.

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**
  - **Problem**: Checking coverage of the descriptions is NP-hard.
- **Second attempt**: Write a simple backtracking algorithm that, given known set of positive and negative results, either says that the whole partial order is covered or gives an uncovered case.

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**
  - **Problem**: Checking coverage of the descriptions is NP-hard.
- **Second attempt**: Write a simple backtracking algorithm that, given known set of positive and negative results, either says that the whole partial order is covered or gives an uncovered case.
  - Worked really well.

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**
  - **Problem**: Checking coverage of the descriptions is NP-hard.
- **Second attempt**: Write a simple backtracking algorithm that, given known set of positive and negative results, either says that the whole partial order is covered or gives an uncovered case.
  - Worked really well.
  - The program guides the research to yet unresolved cases..

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**
  - **Problem**: Checking coverage of the descriptions is NP-hard.
- **Second attempt**: Write a simple backtracking algorithm that, given known set of positive and negative results, either says that the whole partial order is covered or gives an uncovered case.
  - Worked really well.
  - The program guides the research to yet unresolved cases..
  - The input is a description of the partial order and of the current knowledge, so the program can be used for any similar study.

How did we do it?

- How do you manage such a project?
- **First attempt**: Keep track of resolved cases using Google spreadsheet, and check coverage of the descriptions by hand.
  - **Problem**: **MESS**
  - **Problem**: Checking coverage of the descriptions is NP-hard.
- **Second attempt**: Write a simple backtracking algorithm that, given known set of positive and negative results, either says that the whole partial order is covered or gives an uncovered case.
  - Worked really well.
  - The program guides the research to yet unresolved cases..
  - The input is a description of the partial order and of the current knowledge, so the program can be used for any similar study.
  - Source code and input files available as ancillary files in the arxiv version.

Packing a tree into a tree

- Let's recall the poly-time algorithm when H and G are trees.

Packing a tree into a tree

- Let's recall the poly-time algorithm when H and G are trees.
- Take one vertex and guess its image; now we are at subgraph isomorphism of *rooted trees*.

Packing a tree into a tree

- Let's recall the poly-time algorithm when H and G are trees.
- Take one vertex and guess its image; now we are at subgraph isomorphism of *rooted trees*.
- Create dynamic programming table $D[u][v]$, for $u \in V(H)$ and $v \in V(G)$, with the following meaning:

$$D[u][v] = \text{Is there a subgraph isomorphism } \eta \text{ from } H_u \text{ to } G_v \text{ such that } \eta(u) = v?$$

Packing a tree into a tree

- To compute $D[u][v]$, we need to match subtrees of H_u into subtrees of G_v.

Packing a tree into a tree

- To compute $D[u][v]$, we need to match subtrees of H_u into subtrees of G_v.
- This boils down to solving the maximum matching problem in an auxiliary bipartite graph.

Packing a tree into a tree

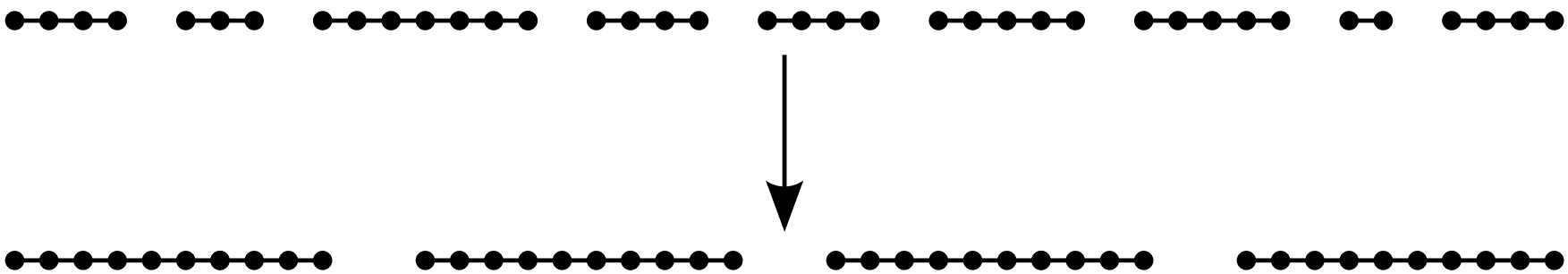
- To compute $D[u][v]$, we need to match subtrees of H_u into subtrees of G_v.
- This boils down to solving the maximum matching problem in an auxiliary bipartite graph.
- Result is $D[r_H][r_G]$.

Packing a forest into a forest

- What if we were packing a forest into a forest?

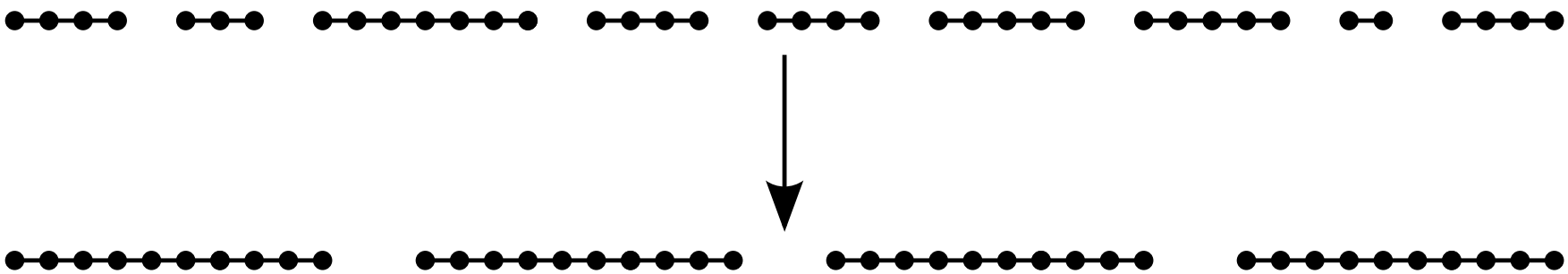
Packing a forest into a forest

- What if we were packing a forest into a forest?
- Bin Packing: Subgraph Isomorphism is already NP-hard for forests of paths!



Packing a forest into a forest

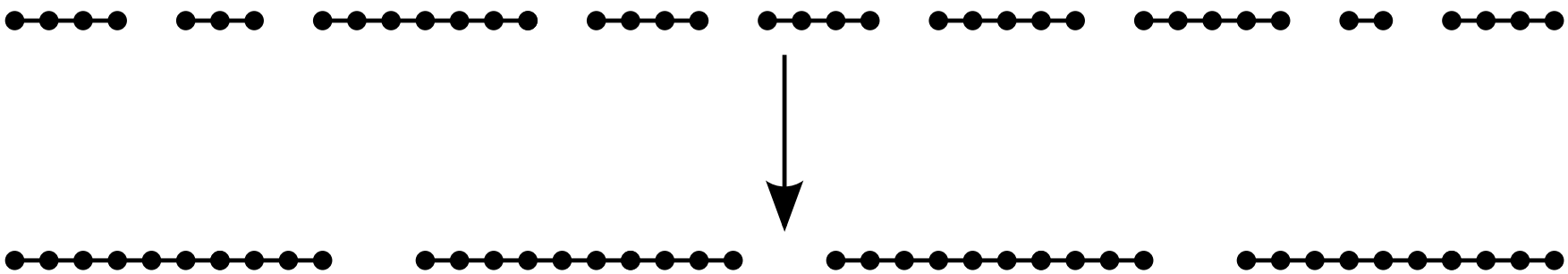
- What if we were packing a forest into a forest?
- Bin Packing: Subgraph Isomorphism is already NP-hard for forests of paths!



- We show an $f(\mathrm{cc}(H)) \cdot n^{\mathcal{O}(1)}$ algorithm for packing a forest into a forest.

Packing a forest into a forest

- What if we were packing a forest into a forest?
- Bin Packing: Subgraph Isomorphism is already NP-hard for forests of paths!



- We show an $f(\mathrm{cc}(H)) \cdot n^{\mathcal{O}(1)}$ algorithm for packing a forest into a forest.
- A very natural case omitted by the previous study (Thm P.8).

Packing a forest into a forest

- For simplicity, let us pack a forest into a tree. Let $k = \mathrm{cc}(H)$.

Packing a forest into a forest

- For simplicity, let us pack a forest into a tree. Let $k = \mathrm{cc}(H)$.
- **Idea**: let's make a similar DP, but now a state consists of:

Packing a forest into a forest

- For simplicity, let us pack a forest into a tree. Let $k = \mathrm{cc}(H)$.
- **Idea**: let's make a similar DP, but now a state consists of:
  - a subtree G_v of G, rooted at v;

Packing a forest into a forest

- For simplicity, let us pack a forest into a tree. Let $k = \mathrm{cc}(H)$.
- **Idea**: let's make a similar DP, but now a state consists of:
  - a subtree G_v of G, rooted at v;
  - a subtree H_u of some connected component of H, rooted at u;

Packing a forest into a forest

- For simplicity, let us pack a forest into a tree. Let $k = \mathrm{cc}(H)$.
- **Idea**: let's make a similar DP, but now a state consists of:
  - a subtree G_v of G, rooted at v;
  - a subtree H_u of some connected component of H, rooted at u;
  - a subset S of the other components of H.

Packing a forest into a forest

- For simplicity, let us pack a forest into a tree. Let $k = \mathrm{cc}(H)$.
- **Idea**: let's make a similar DP, but now a state consists of:
  - a subtree G_v of G, rooted at v;
  - a subtree H_u of some connected component of H, rooted at u;
  - a subset S of the other components of H.
- **Meaning**: $D[u][v][S]$ is true iff the subgraph $H_u \cup S$ can be squeezed into G_v so that u is mapped to v.

Packing a forest into a forest

- Computation of $D[u][v][S]$ involves a variant of maximum matching with $\leq k$ colors.

Packing a forest into a forest

- Computation of $D[u][v][S]$ involves a variant of maximum matching with $\leq k$ colors.
- This variant is NP-hard, but can be solved in FPT time when parameterized by k using algebraic algorithms for matchings.

Packing a forest into a forest

- Computation of $D[u][v][S]$ involves a variant of maximum matching with $\leq k$ colors.
- This variant is NP-hard, but can be solved in FPT time when parameterized by k using algebraic algorithms for matchings.
- The algorithm uses Schwartz-Zippel lemma. Hence it is randomized, and we do not know how to derandomize it.

Feedback vertex set number and planarity

- By Matoušek and Thomas, Subgraph Isomorphism can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for connected H.

Feedback vertex set number and planarity

- By Matoušek and Thomas, Subgraph Isomorphism can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for connected H.
- How much more do we need to assume to remove $\mathrm{tw}(G)$ from the exponent?

Feedback vertex set number and planarity

- By Matoušek and Thomas, Subgraph Isomorphism can be solved in time $f(\Delta(H)) \cdot n^{\mathcal{O}(\mathrm{tw}(G))}$ for connected H.
- How much more do we need to assume to remove $\mathrm{tw}(G)$ from the exponent?
- **Recall**: X is a feedback vertex set if $G \setminus X$ is a forest. $\mathrm{fvs}(G)$ is the minimum size of a fvs in G.

Main algorithmic result

Main algorithmic result, Thm P.10

Subgraph Isomorphism can be solved in time

$$f(\Delta(G), \mathsf{fvs}(G)) \cdot n^{g(\mathsf{cc}(H),\mathsf{genus}(G))}.$$

Main algorithmic result

Main algorithmic result, Thm P.10

Subgraph Isomorphism can be solved in time

$$f(\Delta(G), \mathsf{fvs}(G)) \cdot n^{g(\mathsf{cc}(H),\mathsf{genus}(G))}.$$

- We need to replace $\mathsf{tw}(G)$ with $\mathsf{fvs}(G)$, and assume that G has small degree and very small genus.

Main algorithmic result

Main algorithmic result, Thm P.10

SUBGRAPH ISOMORPHISM can be solved in time

$$f(\Delta(G), \mathrm{fvs}(G)) \cdot n^{g(\mathrm{cc}(H),\mathrm{genus}(G))}.$$

- We need to replace $\mathrm{tw}(G)$ with $\mathrm{fvs}(G)$, and assume that G has small degree and very small genus.
- **Lower bounds**: all these assumptions are necessary!

Glimpse into the proof

- Focus on the case genus$(G) = 0$.

Glimpse into the proof

- Focus on the case genus$(G) = 0$.
- Graphs of small degree and feedback vertex set number have a very special structure:

Glimpse into the proof

- Focus on the case genus$(G) = 0$.
- Graphs of small degree and feedback vertex set number have a very special structure:
  - A simple skeleton with branches sticking out of it.

Glimpse into the proof

- Focus on the case genus(G) $= 0$.
- Graphs of small degree and feedback vertex set number have a very special structure:
  - A simple skeleton with branches sticking out of it.
- Guess how the skeleton of H is embedded into the skeleton of G.

Glimpse into the proof

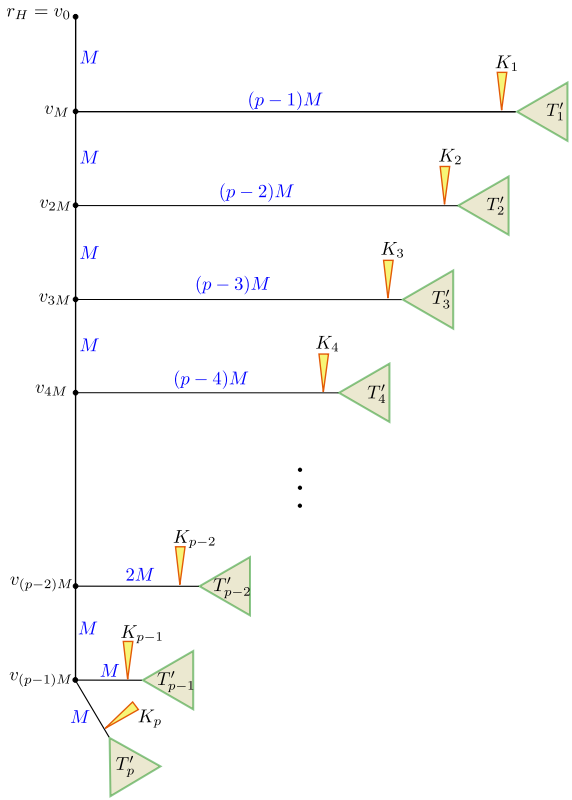
- Focus on the case genus(G) $= 0$.
- Graphs of small degree and feedback vertex set number have a very special structure:
  - A simple skeleton with branches sticking out of it.
- Guess how the skeleton of H is embedded into the skeleton of G.
- Encode the rest as a CSP instance on an *outerplanar graph* that has treewidth 2.

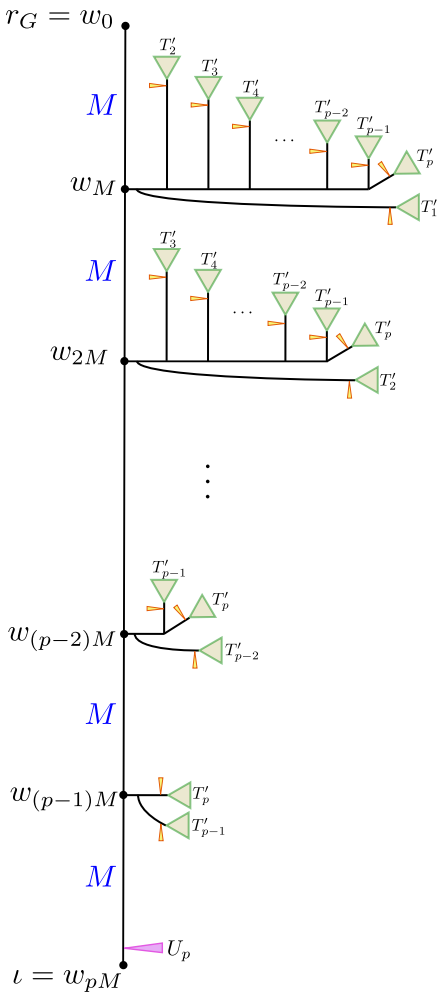
Glimpse into the proof

- Focus on the case genus$(G) = 0$.
- Graphs of small degree and feedback vertex set number have a very special structure:
  - A simple skeleton with branches sticking out of it.
- Guess how the skeleton of H is embedded into the skeleton of G.
- Encode the rest as a CSP instance on an *outerplanar graph* that has treewidth 2.
- For a higher genus g, generalizations of outerplanar graphs have treewidth $\mathcal{O}(g)$.

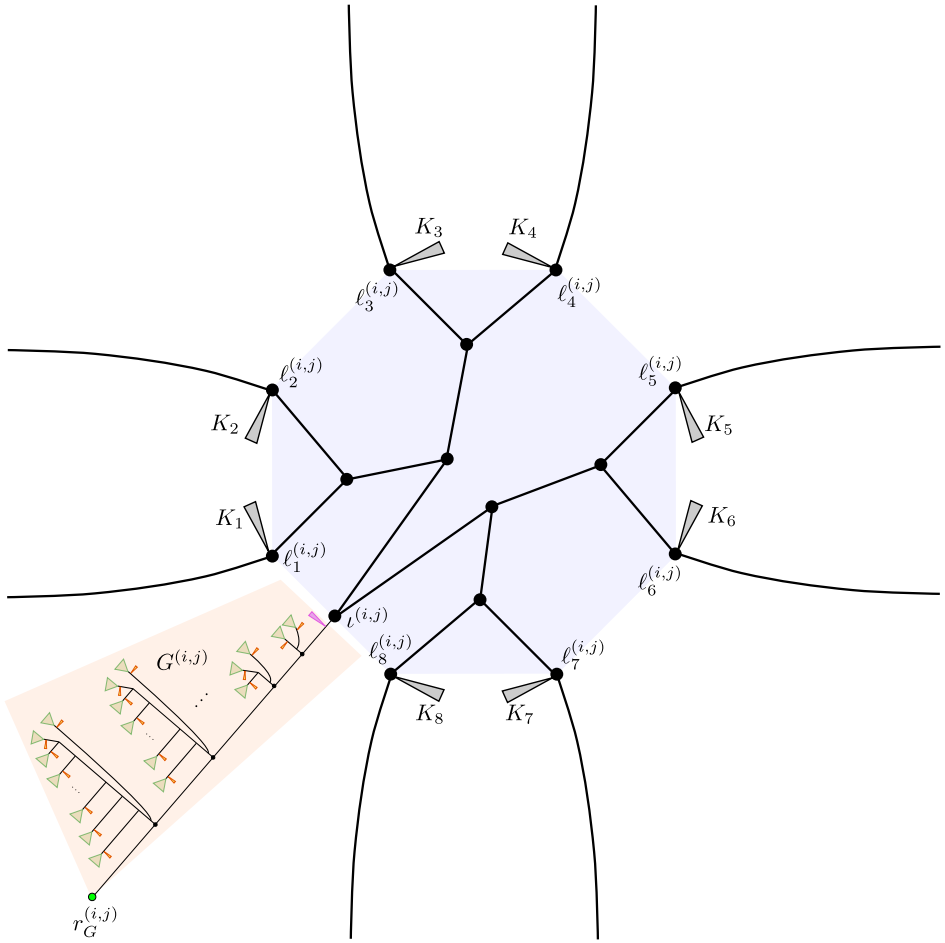
Lower bounds

Lower bounds

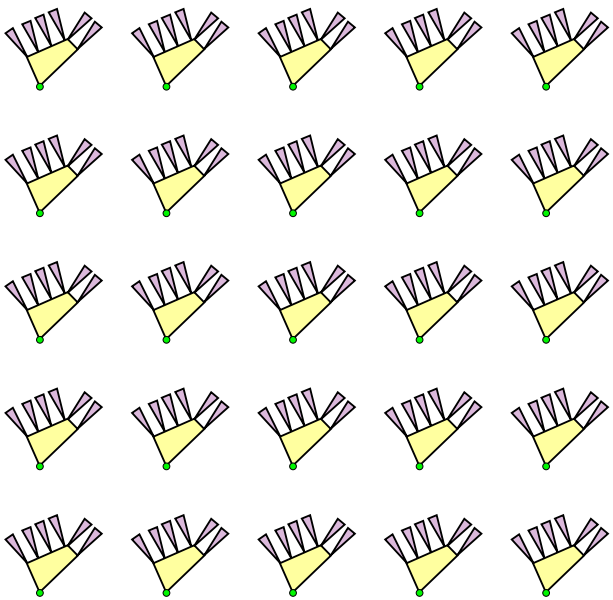


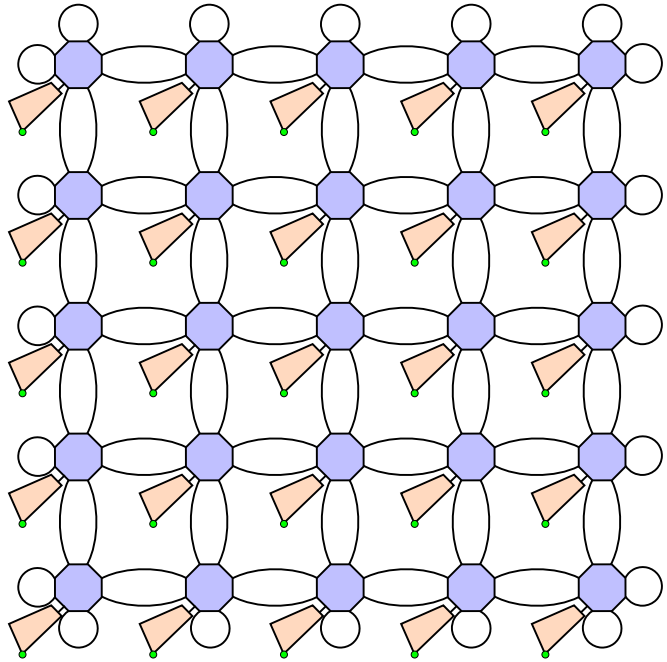


Lower bounds



Lower bounds





Conclusions

- The question you probably ask yourselves...

Conclusions

- The question you probably ask yourselves...
- **WHY?!**

Conclusions

- The question you probably ask yourselves...
- **WHY?!**
- A systematic study can be a managable project even though the number of cases seems hopeless in the beginning.

Conclusions

- The question you probably ask yourselves...
- **WHY?!**
- A systematic study can be a managable project even though the number of cases seems hopeless in the beginning.
- It guides you to new positive cases that would not be discovered otherwise.

Conclusions

- The question you probably ask yourselves...
- **WHY?!**
- A systematic study can be a managable project even though the number of cases seems hopeless in the beginning.
- It guides you to new positive cases that would not be discovered otherwise.
- Subgraph Isomorphism has a particularly rich ecology of parameters, which makes it well-suited for such considerations.

Conclusions

- The question you probably ask yourselves...
- **WHY?!**
- A systematic study can be a managable project even though the number of cases seems hopeless in the beginning.
- It guides you to new positive cases that would not be discovered otherwise.
- Subgraph Isomorphism has a particularly rich ecology of parameters, which makes it well-suited for such considerations.
- **Thanks for attention!**