Everything you always wanted to know about the parameterized complexity of SUBGRAPH ISOMORPHISM

(but were afraid to ask)

Dániel Marx

Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

(Joint work with Michał Pilipczuk)

BWAG: Bertinoro Workshop on Algorithms and Graphs
December 16, 2013
Bertinoro, Italy

SUBGRAPH ISOMORPHISM

Input: Graphs H and G.

Decide: Is H isomorphic to a subgraph of G?

- Hard in general: HAMILTONIAN CYCLE is a special case.
- Hard even for planar graphs and 3-regular graphs.

SUBGRAPH ISOMORPHISM

Input: Graphs H and G.

Decide: Is H isomorphic to a subgraph of G?

- Hard in general: HAMILTONIAN CYCLE is a special case.
- Hard even for planar graphs and 3-regular graphs.

Is SUBGRAPH ISOMORPHISM easy on bounded-treewidth graphs?

DP and bounded treewidth

Standard dynamic programming on a tree decomposition yields the following result:

Fact

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

This algorithm needs that

- H is **small** and
- G has **bounded treewidth.**

Color coding and bounded treewidth

The color coding technique of [Alon, Yuster, Zwick 1994] gives the following algorithm:

Fact

Subgraph Isomorphism can be solved in time $2^{O(|V(H)|)} \cdot n^{O(\mathrm{tw}(H))}$.

This algorithm needs that

- H is **small**,
- H has **bounded treewidth**,

but the treewidth of G can be arbitrary.

Another DP

A dynamic programming algorithm of [Matoušek and Thomas 1992] gives the following result:

Fact

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

This algorithm needs that

- H has **bounded degree**,
- H is **connected**, and
- G has **bounded treewidth**,

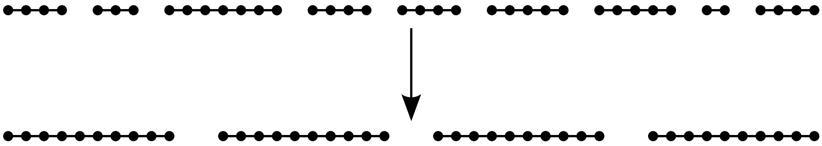
but the size of H can be arbitrary.

Subgraph Isomorphism and Bin Packing

We can reduce Bin Packing (with polynomially bounded sizes) to Subgraph Isomorphism with both H and G being a set of paths:

Fact

Subgraph Isomorphism is NP-hard even if both H and G are sets of paths.



The requirement that H is connected is essential in the [Matoušek and Thomas 1992] result!

Parameters of Subgraph Isomorphism

We have seen that the complexity of Subgraph Isomorphism is influenced by the following parameters of H and G:

- number of vertices,
- treewidth,
- maximum degree,
- number of components (connectedness)

. . . and these parameters can appear either in the exponent of n or as a multiplier of the running time.

Parameters of Subgraph Isomorphism

We have seen that the complexity of Subgraph Isomorphism is influenced by the following parameters of H and G:

- number of vertices,
- treewidth,
- maximum degree,
- number of components (connectedness)

... and these parameters can appear either in the exponent of n or as a multiplier of the running time.

What other natural parameters influence the complexity of Subgraph Isomorphism?

Cliquewidth and Subgraph Isomorphism

Dynamic programming on a tree decomposition:

Fact

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

Can be generalized to clique width [Courcelle, Makowsky, Rotics 2000]:

Fact

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

Cliquewidth and Subgraph Isomorphism

Color coding:

Fact

Subgraph Isomorphism can be solved in time $2^{O(|V(H)|)} \cdot n^{O(\mathrm{tw}(H))}$.

Cannot be generalized to cliquewidth: cliques have cliquewidth 2 and k-Clique is W[1]-hard.

Fact

Subgraph Isomorphism cannot be solved in time $f(|V(H)|) \cdot n^{O(\mathrm{cw}(H))}$ for any computable function f, unless $\mathrm{W[1]} = \mathrm{FPT}$.

Planarity and Subgraph Isomorphism

Fact

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

The result can be generalized to bounded local treewidth and implies the following result for planar G:

Fact

Subgraph Isomorphism for planar G can be solved in time $f(|V(H)|) \cdot n$ for some computable function f.

Planarity and Subgraph Isomorphism

Fact

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

Bounded local treewidth can be further generalized:

Fact

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

- $\mathrm{x}(G)$ is the genus of G,
- $\mathrm{x}(G)$ is the size of the largest clique minor,
- $\mathrm{x}(G)$ is the size of the largest topological clique minor.

Follows from the linear-time solvability of first-order model checking on graphs of bounded expansion [Dvořak, Král, Thomas 2010], [Grohe, Kreutzer, Siebertz 2013].

Treewidth and feedback vertex set number

Fact [Bodlaender 1990], [Ponomarenko 1988]

Graph Isomorphism can be solved in time $n^{O(\mathrm{tw}(G))}$.

Major open question if there is a $f(\mathrm{tw}(G)) \cdot n^{O(1)}$ time algorithm.

Treewidth and feedback vertex set number

Fact [Bodlaender 1990], [Ponomarenko 1988]

Graph Isomorphism can be solved in time $n^{O(\mathrm{tw}(G))}$.

Major open question if there is a $f(\mathrm{tw}(G)) \cdot n^{O(1)}$ time algorithm.

Feedback vertex set number fvs(G): minimum number of vertices whose deletion makes the graph a forest.

Easy: $\mathrm{tw}(G) \leq \mathrm{fvs}(G) + 1$.

Fact [Kratsch and Schweitzer 2010]

Graph Isomorphism can be solved in time $f(\mathrm{fvs}(G)) \cdot n^{O(1)}$.

Is feedback vertex set number a relevant parameter also for Subgraph Isomorphism?

Parameters

We consider the following 10 parameters for H and G:

1. Number of vertices $|V(\cdot)|$.
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. Clique width $\mathrm{cw}(\cdot)$.
8. Genus $\mathrm{genus}(\cdot)$.
9. Hadwiger number (largest clique minor) $\mathrm{hadw}(\cdot)$.
10. Topological Hadwiger number (largest topological clique minor) $\mathrm{hadw}_T(\cdot)$.

Main result

Goal

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

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

Main result

Goal

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

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

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.

Main result

Goal

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

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

Main result

For any combination of the 2×10 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 standard complexity assumption).

Main result

Goal

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

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

Main result

For any combination of the 2×10 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 standard complexity assumption).

Every question in this framework is completly answered by

- 11 positive results and
- 17 negative results.

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 → 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.

Comparing specifications

Finding an algorithm satisfying a specification does not become any easier if we

- remove a parameter,
- move a parameter from the exponent to the multiplier,
- remove a constraint,
- adding a parameter to the multiplier or the exponent whose value is already bounded by a constraint, or
- adding a parameter to the multiplier (resp., exponent) whose value can be bounded by a computable function of the parameters already in the multiplier (resp., exponent) on instances where all the constraints in the description hold.

Comparing specifications

Finding an algorithm satisfying a specification does not become any easier if we

- remove a parameter,
- move a parameter from the exponent to the multiplier,
- remove a constraint,
- adding a parameter to the multiplier or the exponent whose value is already bounded by a constraint, or
- adding a parameter to the multiplier (resp., exponent) whose value can be bounded by a computable function of the parameters already in the multiplier (resp., exponent) on instances where all the constraints in the description hold.

Claim

For any specification, the 11 positive and 17 negative results imply a positive or negative answer using these rules.

Can be verified by a computer program.

Finding a tree in a tree

Fact

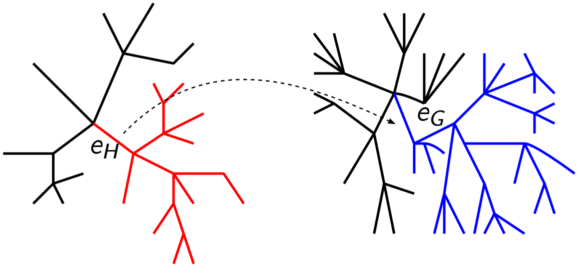
SUBGRAPH ISOMORPHISM is polynomial-time solvable if both H and G are trees.

Finding a tree in a tree

Fact

Subgraph Isomorphism is polynomial-time solvable if both H and G are trees.

Dynamic programming: for every edge-defined subtree $T_H \subseteq H$ and $T_G \subseteq G$, we let $x(T_H, T_G) = \text{true}$ if there is a subgraph isomorphism from T_H to T_G matching the root edges.

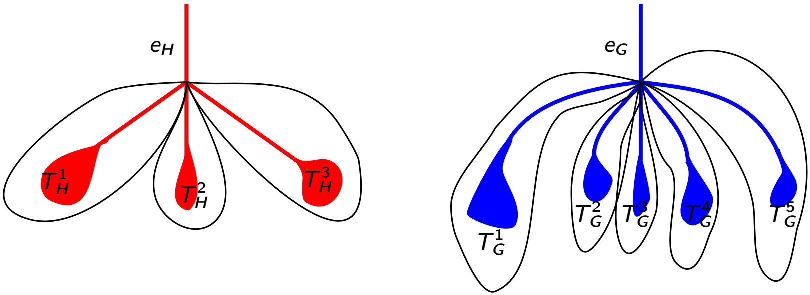


Finding a tree in a tree

Fact

SUBGRAPH ISOMORPHISM is polynomial-time solvable if both H and G are trees.

We need to match the children trees of T_H to the children trees of T_G: we need to solve a bipartite matching problem.

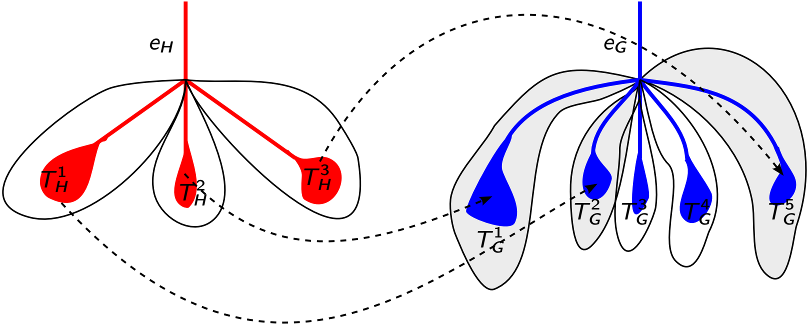


Finding a tree in a tree

Fact

Subgraph Isomorphism is polynomial-time solvable if both H and G are trees.

We need to match the children trees of T_H to the children trees of T_G: we need to solve a bipartite matching problem.



Finding a forest in a tree

Positive result P.8

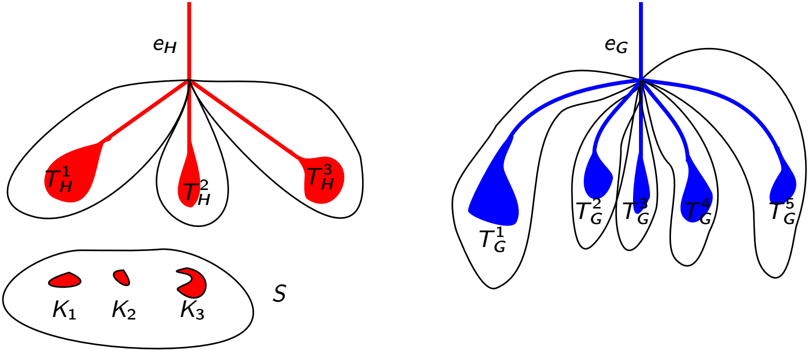
SUBGRAPH ISOMORPHISM can be solved in randomized time $f(\mathrm{cc}(H)) \cdot n^{O(1)}$ if G is a tree.

Finding a forest in a tree

Positive result P.8

SUBGRAPH ISOMORPHISM can be solved in randomized time $f(\mathrm{cc}(H)) \cdot n^{O(1)}$ if G is a tree.

Dynamic programming: for every pair of edge-defined subtrees $T_H \subseteq H$, $T_G \subseteq H$, and subset S of components of H, we let $x(T_H, T_G, S) = \text{true}$ if there is a subgraph isomorphism from $T_H \cup S$ to T_G matching the root edges.

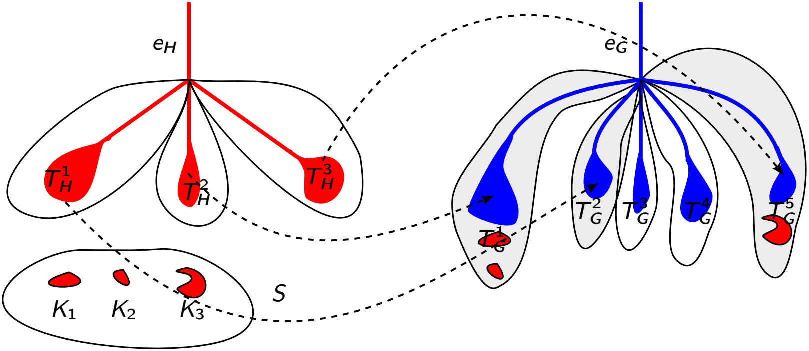


Finding a forest in a tree

Positive result P.8

SUBGRAPH ISOMORPHISM can be solved in randomized time $f(\mathrm{cc}(H)) \cdot n^{O(1)}$ if G is a tree.

Dynamic programming: for every pair of edge-defined subtrees $T_H \subseteq H$, $T_G \subseteq H$, and subset S of components of H, we let $x(T_H, T_G, S) = \text{true}$ if there is a subgraph isomorphism from $T_H \cup S$ to T_G matching the root edges.



Finding a forest in a tree

Essentially, we need to solve bipartite (perfect) matching with an additional restriction: the edges are labeled with ℓ colors and we need **at least one** edge of each color.

Finding a forest in a tree

Essentially, we need to solve bipartite (perfect) matching with an additional restriction: the edges are labeled with ℓ colors and we need **at least one** edge of each color.

Fact [Mulmuley, Vazirani, Vazirani 1987]

Given a bipartite (multi)graph B with nonnegative integer weights and a target weight w, there is a randomized algorithm for finding a perfect matching of weight **exactly** w in time polynomial in $|B|$ and w.

Finding a forest in a tree

Essentially, we need to solve bipartite (perfect) matching with an additional restriction: the edges are labeled with ℓ colors and we need **at least one** edge of each color.

Fact [Mulmuley, Vazirani, Vazirani 1987]

Given a bipartite (multi)graph B with nonnegative integer weights and a target weight w, there is a randomized algorithm for finding a perfect matching of weight **exactly** w in time polynomial in $|B|$ and w.

We can solve our colored matching problem in time $f(\ell) \cdot n^{O(1)}$:

- Replace each edge of label i with two parallel edges of weight 0 and $2^{i-1} + 2^{2\ell-i}$.
- Find a perfect matching of weight exactly $w = \sum_{i=1}^{\ell}(2^{i-1} + 2^{2\ell-i})$.

— intermission —

Constraint satisfaction problems

We define a Constraint Satisfaction Problem (CSP) by

- a domain D of values,
- a set V of variables, and
- a set of constraints, where a constraint is a binary relation on two variables.

Examples:

- 3-Coloring of G is a CSP with $|D| = 3$ and $V = V(G)$,
- k-Coloring of G is a CSP with $D = V(G)$ and $|V| = k$.

Constraint satisfaction problems

We define a Constraint Satisfaction Problem (CSP) by

- a domain D of values,
- a set V of variables, and
- a set of constraints, where a constraint is a binary relation on two variables.

Examples:

- 3-Coloring of G is a CSP with $|D| = 3$ and $V = V(G)$,
- k-Coloring of G is a CSP with $D = V(G)$ and $|V| = k$.

Primal graph: vertex set is V, there is an edge between u and v if there is a constraint on u and v.

Fact [Freuder 1990]

A CSP instance with primal graph G can be solved in time $n^{O(\mathrm{tw}(G))}$.

Projections

Projection graph: vertex set is V, there is an edge $\overrightarrow{uv}$ if the constraint on u and v is a projection from u to v.

- a **projection source** makes the problem easy to solve.
- a **projection sink** is useless in general.

Projections

Projection graph: vertex set is V, there is an edge $\overrightarrow{uv}$ if the constraint on u and v is a projection from u to v.

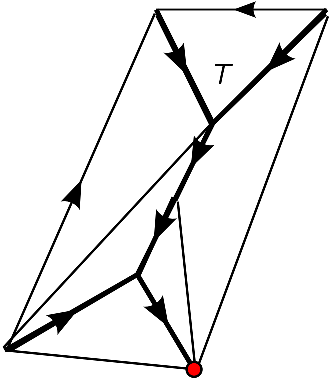
- a **projection source** makes the problem easy to solve.
- a **projection sink** is useless in general.

Fact

We can solve in polynomial time a CSP instance if its primal graph is planar and has a projection sink.

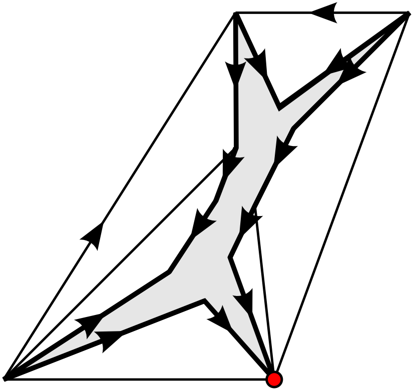
Projections

- There is a spanning in-tree T rooted at the projection sink.



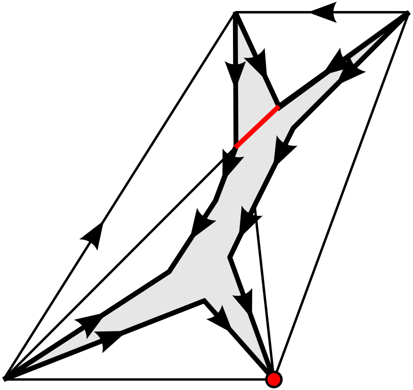
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.



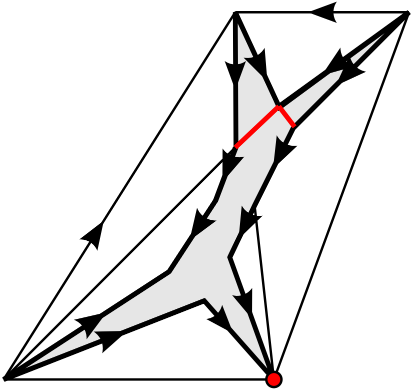
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.



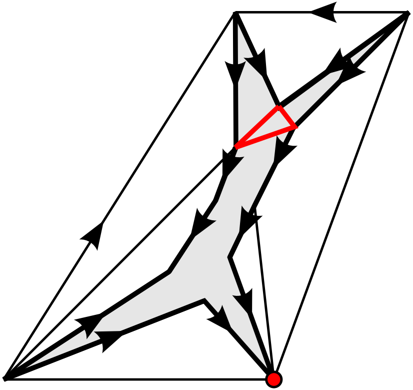
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.



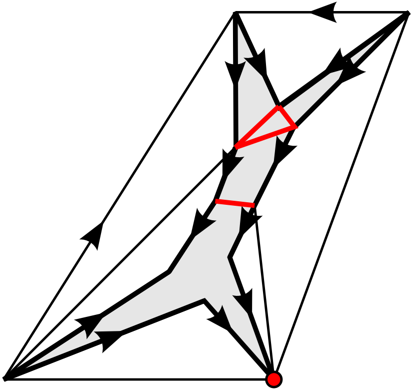
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.



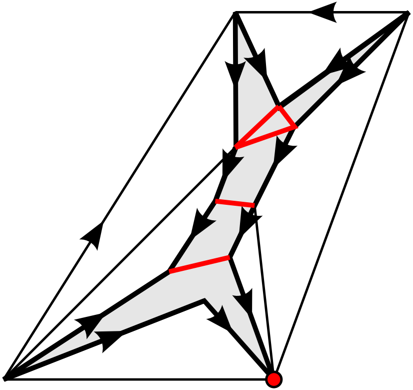
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.



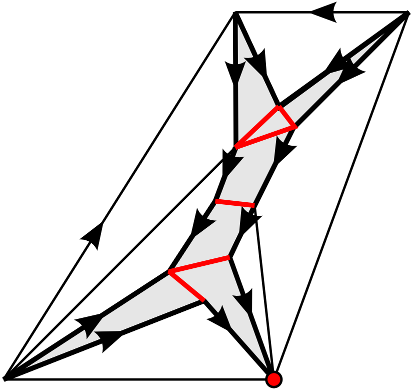
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.



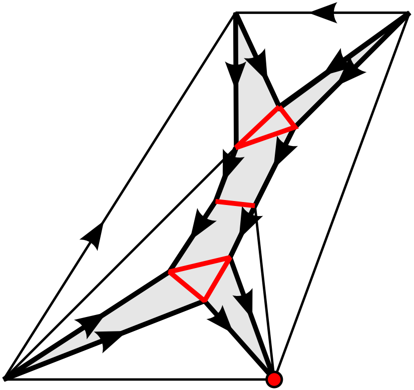
Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.



Projections

- There is a spanning in-tree T rooted at the projection sink.
- Cut open this tree, duplicating variables and constraints the obvious way.
- Duplicated variables are automatically synchronized.
- Resulting primal graph is outerplanar $\Rightarrow$ has treewidth $\leq 3 \Rightarrow$ polynomial-time solvable.



— end of intermission —

Feedback vertex set number

Positive result

Subgraph Isomorphism can be solved in time $f(\mathrm{fvs}(G), \Delta(G)) \cdot n^{O(1)}$ if G is planar and H is connected.

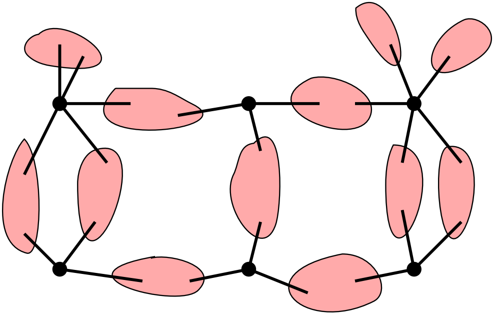
Feedback vertex set number

Positive result

Subgraph Isomorphism can be solved in time $f(\mathrm{fvs}(G), \Delta(G)) \cdot n^{O(1)}$ if G is planar and H is connected.

Fact

There is a set Z of $O(\Delta^2(G) \cdot \mathrm{fvs}(G))$ vertices such that every component of $G \setminus Z$ is a tree and has at most two edges to Z.



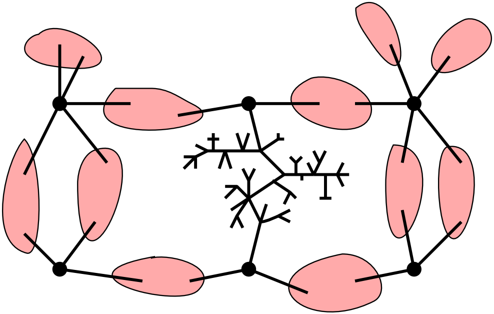
Feedback vertex set number

Positive result

Subgraph Isomorphism can be solved in time $f(\mathrm{fvs}(G), \Delta(G)) \cdot n^{O(1)}$ if G is planar and H is connected.

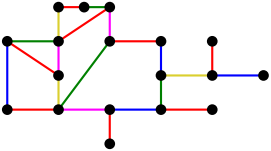
Fact

There is a set Z of $O(\Delta^2(G) \cdot \mathrm{fvs}(G))$ vertices such that every component of $G \setminus Z$ is a tree and has at most two edges to Z.

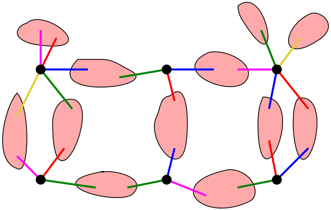


Feedback vertex set number

- Let us guess which subset of edges indicident to Z is used by the solution.
- Let us fix an edge coloring of H and let us guess the correct edge coloring of the edges incident to Z.



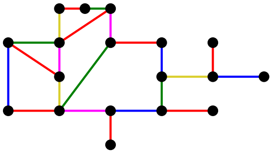
H



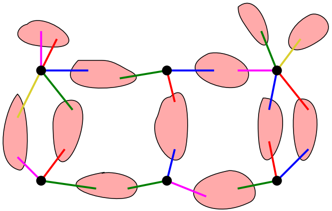
G

Feedback vertex set number

- We formulate finding the subgraph isomorphism $\phi : V(H) \to V(G)$ as a CSP problem: the variables correspond to the vertices in Z and the value of a $u \in Z$ is the preimage $\phi^{-1}(u)$.



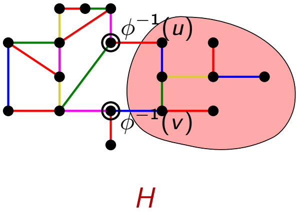
H

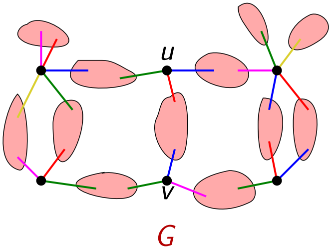


G

Feedback vertex set number

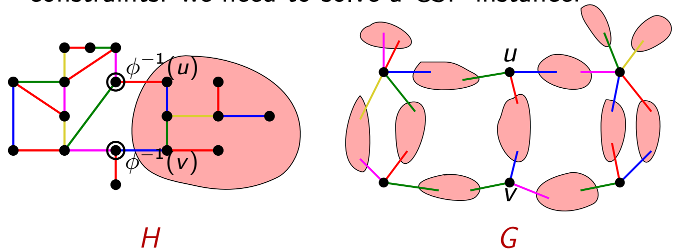
- We formulate finding the subgraph isomorphism $\phi : V(H) \to V(G)$ as a CSP problem: the variables correspond to the vertices in Z and the value of a $u \in Z$ is the preimage $\phi^{-1}(u)$.
- If $G \setminus Z$ has a component adjacent to $u, v \in Z$, then this forces a constraint on $\phi^{-1}(u)$ and $\phi^{-1}(v)$.





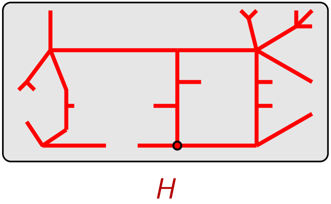
Feedback vertex set number

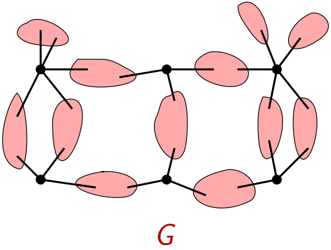
- We formulate finding the subgraph isomorphism $\phi : V(H) \to V(G)$ as a CSP problem: the variables correspond to the vertices in Z and the value of a $u \in Z$ is the preimage $\phi^{-1}(u)$.
- If $G \setminus Z$ has a component adjacent to $u, v \in Z$, then this forces a constraint on $\phi^{-1}(u)$ and $\phi^{-1}(v)$.
- After taking care of some technicalities, these are the only constraints: we need to solve a CSP instance.



Feedback vertex set number

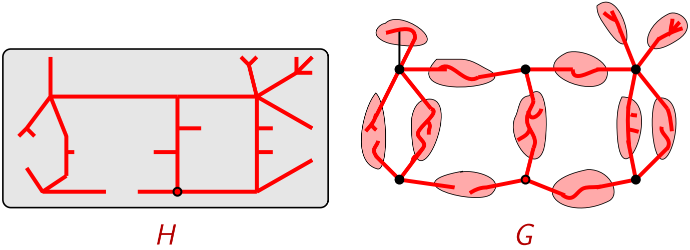
- Fix a spanning tree in H and guess how the tree goes via the components of $G \setminus Z$ (which components are traversed).





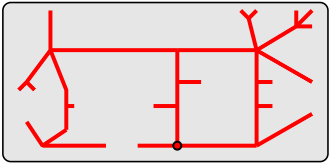
Feedback vertex set number

- Fix a spanning tree in H and guess how the tree goes via the components of $G \setminus Z$ (which components are traversed).

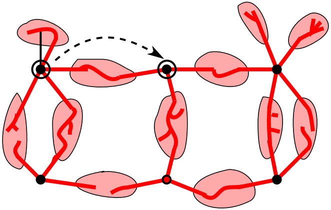


Feedback vertex set number

- Fix a spanning tree in H and guess how the tree goes via the components of $G \setminus Z$ (which components are traversed).
- This forces some constraints to be projections, in fact, creates a projection sink.
- As G is planar, the primal graph is planar and the CSP instance can be solved in polynomial time.



H



G

Feedback vertex set number

Positive result

SUBGRAPH ISOMORPHISM can be solved in time $f(\mathrm{fvs}(G), \Delta(G)) \cdot n^{O(1)}$ if G is planar and H is connected.

Feedback vertex set number

Positive result

Subgraph Isomorphism can be solved in time $f(\mathrm{fvs}(G), \Delta(G)) \cdot n^{O(1)}$ if G is planar and H is connected.

Can be generalized:

Positive result P.10

Subgraph Isomorphism can be solved in time $f_1(\mathrm{fvs}(G), \Delta(G)) \cdot n^{f_2(\mathrm{genus}(G), \mathrm{cc}(H))}$.

Positive result P.11

Subgraph Isomorphism can be solved in time $f_1(\mathrm{fvs}(G), \Delta(G)) \cdot n^{f_2(\mathrm{hadw}(G), \mathrm{cc}(H), \Delta(H))}$.

Graph Structure Theorem

Decomposing H-minor-free graphs into almost embeddable parts:

Theorem [Robertson-Seymour]

For every graph H, there is an integer k and a surface Σ such that every H-minor-free graph

- can be built by clique sums from graphs that are k-almost embeddable in Σ,

(or equivalently)

- has a tree decomposition where every torso is k-almost embeddable in Σ.

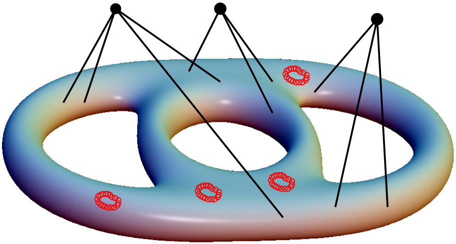
Originally stated only combinatorially, algorithmic versions are known.

k-almost embeddable

Definition

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

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



Projection sinks

Fact

We can solve in polynomial time a CSP instance if its primal graph G is planar and has a projection sink.

Straightforward generalization:

Fact

We can solve in time $n^{f(\text{genus}(G))}$ a CSP instance if its primal graph G has a projection sink.

Projection sinks

Fact

We can solve in polynomial time a CSP instance if its primal graph G is planar and has a projection sink.

Straightforward generalization:

Fact

We can solve in time $n^{f(\text{genus}(G))}$ a CSP instance if its primal graph G has a projection sink.

By using the Robertson-Seymour structure theorem and carefully handling vortices and cliques sums, we get the following generalization:

Fact

We can solve in time $n^{f(\text{hadw}(G))}$ a CSP instance if its primal graph G has a projection sink.

Hardness proofs

To prove that there is no algorithm for Subgraph Isomorphism with running time

$$n^{f(p_1,\ldots,p_t)},$$

we should show that Subgraph Isomorphism is NP-hard even for instances where each of $p_1, \ldots, p_t$ is bounded by a universal constant.

Hardness proofs

To prove that there is no algorithm for Subgraph Isomorphism with running time

$$n^{f(p_1,\ldots,p_t)},$$

we should show that Subgraph Isomorphism is NP-hard even for instances where each of $p_1, \ldots, p_t$ is bounded by a universal constant.

To prove that there is no algorithm for Subgraph Isomorphism with running time

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

we use the fact that there is no $f(k) \cdot n^{O(1)}$ algorithm for k-Clique unless $\mathrm{FPT} = \mathrm{W}[1]$. Then we need a reduction from k-Clique to Subgraph Isomorphism such that

- each of $p_1, \ldots, p_\ell$ is bounded by a function of k, and
- each of $p_{\ell+1}, \ldots, p_t$ is bounded by a universal constant.

Grid Tiling

Grid Tiling

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Horizontal neighbors agree in the first component.
- Vertical neighbors agree in the second component.

(1,1) (1,3) (4,2)	(1,5) (4,1) (3,5)	(1,1) (4,2) (3,3)
(2,2) (4,1)	(1,3) (2,1)	(2,2) (3,2)
(3,1) (3,2) (3,3)	(1,1) (3,1)	(3,2) (3,5)

$k = 3, \ D = 5$

Grid Tiling

Grid Tiling

Input: A $k \times k$ matrix and a set of pairs $S_{i,j} \subseteq [D] \times [D]$ for each cell.

Find: A pair $s_{i,j} \in S_{i,j}$ for each cell such that

- Horizontal neighbors agree in the first component.
- Vertical neighbors agree in the second component.

(1,1) (1,3) (4,2)	(1,5) (4,1) (3,5)	(1,1) (4,2) (3,3)
(2,2) (4,1)	(1,3) (2,1)	(2,2) (3,2)
(3,1) (3,2) (3,3)	(1,1) (3,1)	(3,2) (3,5)

$k = 3,\ D = 5$

Grid Tiling

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	(v_i, v_i)			

Each diagonal cell defines a value v_i . . .

Grid Tiling

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(., v_i)$			
$(v_i, .)$	(v_i, v_i)	$(v_i, .)$	$(v_i, .)$	$(v_i, .)$
	$(., v_i)$			
	$(., v_i)$			
	$(., v_i)$			

... which appears on a "cross"

Grid Tiling

Reduction from k-CLIQUE

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(., v_i)$			
$(v_i, .)$	(v_i, v_i)	$(v_i, .)$	$(v_i, .)$	$(v_i, .)$
	$(., v_i)$			
	$(., v_i)$		(v_j, v_j)	
	$(., v_i)$			

v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

Grid Tiling

Reduction from k-Clique

Definition of the sets:

- For $i = j$: $(x, y) \in S_{i,j} \iff x = y$
- For $i \neq j$: $(x, y) \in S_{i,j} \iff x$ and y are adjacent.

	$(., v_i)$		$(., v_j)$	
$(v_i, .)$	(v_i, v_i)	$(v_i, .)$	(v_i, v_j)	$(v_i, .)$
	$(., v_i)$		$(., v_j)$	
$(v_j, .)$	(v_j, v_i)	$(v_j, .)$	(v_j, v_j)	$(v_j, .)$
	$(., v_i)$		$(., v_j)$	

v_i and v_j are adjacent for every $1 \leq i < j \leq k$.

Hardness proofs

Negative result N.7

Unless $\mathrm{FPT} = \mathrm{W}[1]$, there is no algorithm for SUBGRAPH ISOMORPHISM with running time

$$f_1(\mathrm{cc}(H), \mathrm{pw}(G), \mathrm{fvs}(G)) \cdot n^{f_2(\mathrm{pw}(H))},$$

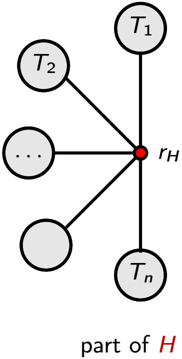
even if H is a forest and G is a connected planar graph with maximum degree 3.

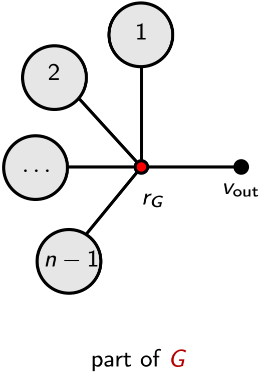
We need to reduce $k \times k$ GRID TILING to an instance of SUBGRAPH ISOMORPHISM where

- $\mathrm{cc}(H)$, $\mathrm{pw}(G)$, $\mathrm{fvs}(G)$ is bounded by a function of k,
- $\mathrm{pw}(H)$ is bounded by a universal constant,
- H is a forest,
- G is a connected planar graph with maximum degree 3.

Gadget construction

Consider the following subgraphs in H and G:

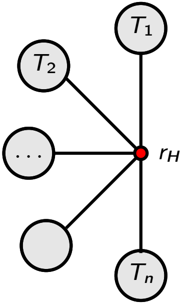




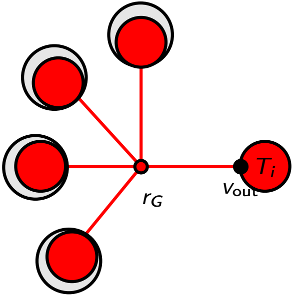
If a subgraph isomorphism maps special vertex r_H to r_G, then one of the tree T_i protrudes out at vertex v_{out}.

Gadget construction

Consider the following subgraphs in H and G:



part of H

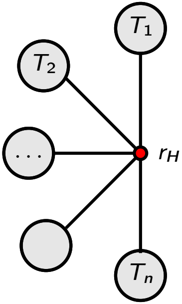


part of G

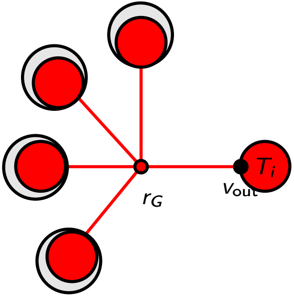
If a subgraph isomorphism maps special vertex r_H to r_G, then one of the tree T_i protrudes out at vertex v_{out}.

Gadget construction

Consider the following subgraphs in H and G:



part of H



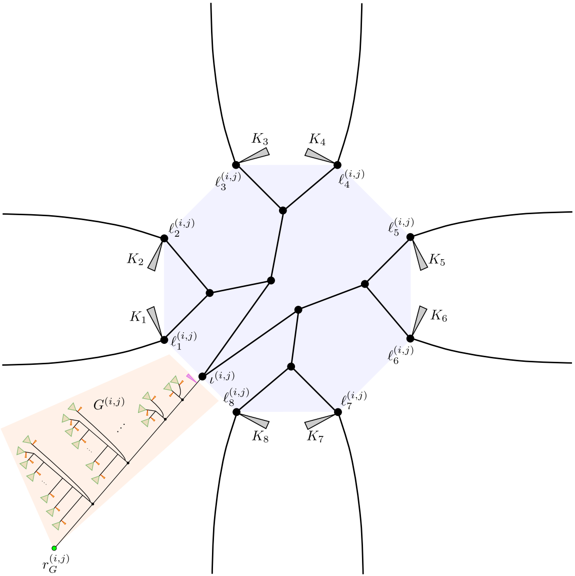
part of G

If a subgraph isomorphism maps special vertex r_H to r_G, then one of the tree T_i protrudes out at vertex v_{out}.

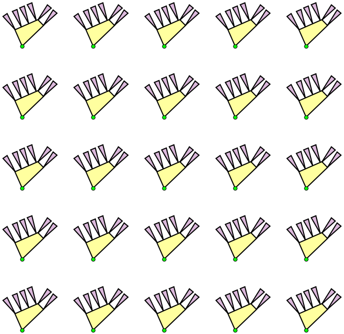
Slightly more challenging: construct such gadgets where H has bounded degree and bounded pathwidth.

Hardness proofs

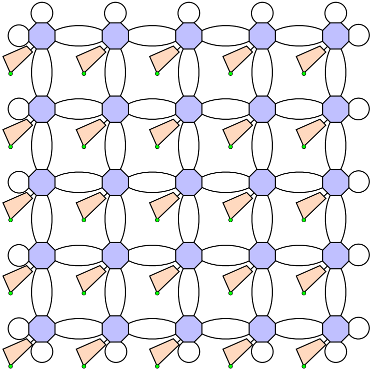
Generalizing it to a gadget where 8 paths of certain lengths protrude out:



Hardness proofs



Graph *H*



Graph *G*

Hardness proofs

Negative result N.7

Unless $\mathsf{FPT} = \mathsf{W}[1]$, there is no algorithm for Subgraph Isomorphism with running time

$$f_1(\mathrm{cc}(H), \mathrm{pw}(G), \mathrm{fvs}(G)) \cdot n^{f_2(\mathrm{pw}(H))},$$

even if H is a forest and G is a connected planar graph with maximum degree 3.

Hardness proofs

Negative result N.7

Unless FPT = W[1], there is no algorithm for Subgraph Isomorphism with running time

$$f_1(\mathrm{cc}(H), \mathrm{pw}(G), \mathrm{fvs}(G)) \cdot n^{f_2(\mathrm{pw}(H))},$$

even if H is a forest and G is a connected planar graph with maximum degree 3.

A variant of the result:

Negative result N.8

Unless FPT = W[1], there is no algorithm for Subgraph Isomorphism with running time

$$f_1(\cancel{\mathrm{cc}(H)}, \mathrm{pw}(G), \mathrm{fvs}(G), \mathrm{genus}(G), \Delta(G)) \cdot n^{f_2(\mathrm{pw}(H), \mathrm{hadw}(G))},$$

even if H is a ~~forest~~ tree and ~~G is a connected planar graph with maximum degree 3~~.

Essentially: add a new vertex to H and G to make them connected.

Hardness proofs

Negative result N.7

Unless $\mathsf{FPT} = \mathsf{W}[1]$, there is no algorithm for SUBGRAPH ISOMORPHISM with running time

$$f_1(\mathrm{cc}(H), \mathrm{pw}(G), \mathrm{fvs}(G)) \cdot n^{f_2(\mathrm{pw}(H))},$$

even if H is a forest and G is a connected planar graph with maximum degree 3.

Hardness proofs

Negative result N.7

Unless FPT = W[1], there is no algorithm for Subgraph Isomorphism with running time

$$f_1(\mathrm{cc}(H), \mathrm{pw}(G), \mathrm{fvs}(G)) \cdot n^{f_2(\mathrm{pw}(H))},$$

even if H is a forest and G is a connected planar graph with maximum degree 3.

A variant of the result:

Negative result N.9

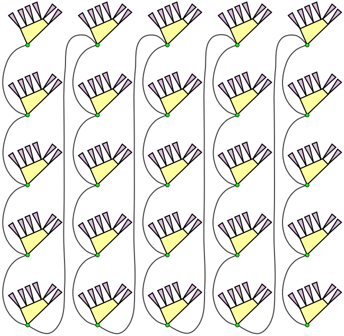
Unless FPT = W[1], there is no algorithm for Subgraph Isomorphism with running time

$$f_1(\cancel{\mathrm{cc}(H)}, \mathrm{pw}(G), \mathrm{fvs}(G), \mathrm{genus}(G)) \cdot n^{f_2(\mathrm{pw}(H))},$$

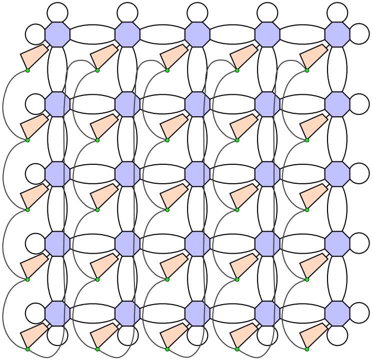
even if H is a ~~forest~~ tree and G is a connected ~~planar~~ graph with maximum degree 3.

Essentially: connecting the gadgets in a path like manner.

Hardness proofs



Graph H



Graph G

Summary

- We formulated a framework with 2×10 parameters and 5 constraints.
- We showed that 11 positive results and 17 negative results (some known, some new) answer every question in this framework.
- We developed a computer program to check for complete coverage and to find the questions that are not yet explained by the results.
- Some interesting new positive results and very careful and nontrivial hardness proofs.
- Full paper and program on arxiv.