Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and Turing kernels

Dániel Marx[1] Bart M.P. Jansen[2]

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

[2]Eindhoven University of Technology,
The Netherlands

WorKer 2015
Nordfjordeid, Norway
June 1, 2015

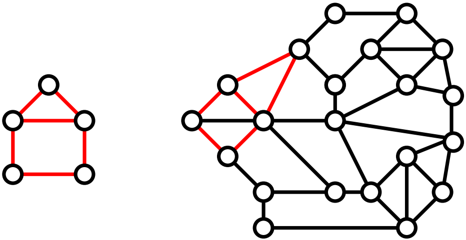
Subgraph Isomorphism

Subgraph Isomorphism

Input: two graphs H and G.

Parameter: $|V(H)|$

Task: decide if G has a subgraph isomorphic to H.

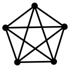
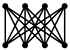
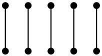
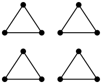


Pattern H Host G

For a class $\mathcal{F}$ of graphs, $\mathcal{F}$-Subgraph Isomorphism is the restriction of the problem when the pattern H is in $\mathcal{F}$.

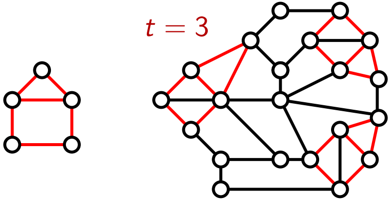
Special cases of Subgraph Isomorphism

We can express the following well-studied problems as special cases of Subgraph Isomorphism:

	Clique NP-hard, W[1]-hard
	Biclique NP-hard, W[1]-hard
	Long Path NP-hard, FPT, no polynomial kernel unless NP $\subseteq$ coNP/poly.
	Matching Polynomial-time solvable.
	Triangle Packing NP-hard, FPT, has a polynomial kernel.

H-packing

Packing	
Input:	two graphs H and G, an integer t.
Parameter:	$t \cdot \lvert V(H) \rvert$
Task:	decide if there are t vertex-disjoint subgraphs of G, each isomorphic to H.



Pattern H Host G

- For a fixed graph H, H-Packing is the problem restricted to a fixed pattern graph H.
- For a class $\mathcal{F}$ of graphs, $\mathcal{F}$-Packing is the restriction of the problem when the pattern H is in $\mathcal{F}$.

Main goal

Question

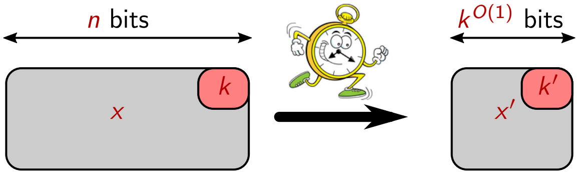
What kind of pattern graphs make Packing and Subgraph Isomorphism easy?

- Formally, characterize the classes $\mathcal{F}$ for which these problems have
  - polynomial-time algorithms,
  - polynomial kernels,
  - polynomial Turing kernels.
- Our goal is to prove dichotomy theorems: the problem is easy if and only if $\mathcal{F}$ has certain property, and hard otherwise.
- To make this technically feasible, we focus on *hereditary* classes: we assume that $\mathcal{F}$ is closed under taking induced subgraphs.

Many-one vs. Turing kernels

Polynomial many-one kernels

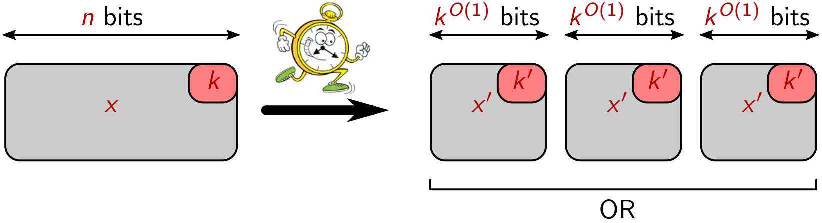
Given an instance (x, k), creates an equivalent instance (x', k') with $|x'| = k^{O(1)}$ and $k' = k^{O(1)}$ in time $(|x| + k)^{O(1)}$.



Many-one vs. Turing kernels

Polynomial Turing kernels

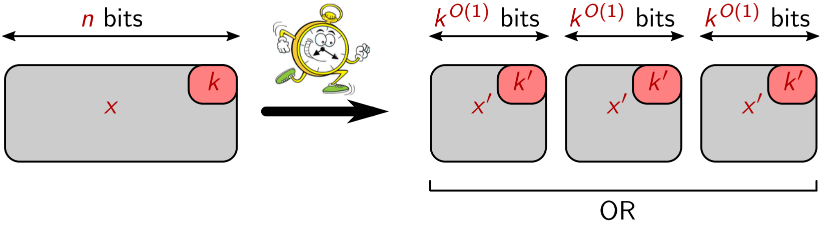
Solves instance (x, k) in time $(|x| + k)^{O(1)}$ using oracle access solving instances (x', k') with $|x'| = k^{O(1)}$ and $k' = k^{O(1)}$ in a single step.



Many-one vs. Turing kernels

Polynomial Turing kernels

Solves instance (x, k) in time $(|x|+k)^{O(1)}$ using oracle access solving instances (x', k') with $|x'| = k^{O(1)}$ and $k' = k^{O(1)}$ in a single step.



- Most typical form: it creates $|x|^{O(1)}$ instances such that the answer is the OR of these instances.
- Negative evidence for polynomial Turing kernels: WK[1]-hardness introduced by [Hermelin et al. 2013].

Packing

Polynomial-time solvability is well-understood:

Theorem [Kirkpatrick and Hell 1978]

H-Packing is NP-hard for every connected graph H with at least 3 vertices.

Packing

Polynomial-time solvability is well-understood:

Theorem [Kirkpatrick and Hell 1978]

H-Packing is NP-hard for every connected graph H with at least 3 vertices.

Easy extensions to disconnected graphs and graph classes:

Corollary

H-Packing is polynomial-time solvable if every component of H has at most two vertices, and NP-hard otherwise.

Corollary

$\mathcal{F}$-Packing is polynomial-time solvable if every component of every graph in $\mathcal{F}$ has at most two vertices, and NP-hard otherwise.

Packing

Kernelization is also well understood:

- For every fixed H, there is a kernel of size $O(k^{|V(H)|})$.
- Interpret the problem as packing of sets of size $|V(H)|$, then kernelization using the Sunflower Lemma.

Packing

Kernelization is also well understood:

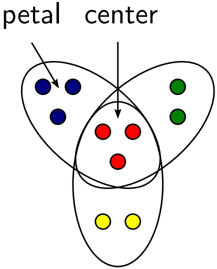
- For every fixed H, there is a kernel of size $O(k^{|V(H)|})$.
- Interpret the problem as packing of sets of size $|V(H)|$, then kernelization using the Sunflower Lemma.

Better question: pattern H is part of the input, but restricted to a class $\mathcal{F}$.

But before that, a short recap...

Sunflower lemma

Definition: Sets $S_1, S_2, \ldots, S_k$ form a **sunflower** if the sets $S_i \setminus (S_1 \cap S_2 \cap \cdots \cap S_k)$ are disjoint.



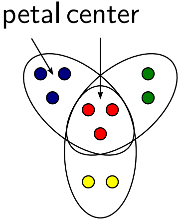
Sunflower Lemma [Erdős and Rado, 1960]

If the size of a set system is greater than $(p-1)^d \cdot d!$ and it contains only sets of size at most d, then the system contains a sunflower with p petals. Furthermore, in this case such a sunflower can be found in polynomial time.

Sunflowers and packing

d-Set Packing

Given a collection $\mathcal{S}$ of sets of size at most d and an integer t, find a set S of t elements that intersects every set of $\mathcal{S}$.



Reduction Rule

Suppose more than $dt+1$ sets form a sunflower.

- If the sets are disjoint $\Rightarrow$ we are done.
- Otherwise, keep only $dt+1$ of the sets.

Marking

Another interpretation:

We can mark a set M of $f(d)t^d$ elements such that the following holds. If Z is any set of at most dt elements and there is an $S \in \mathcal{S}$ with $S \cap Z = \emptyset$, then there is also such an $S \subseteq M$.

$$\Downarrow$$

We can mark a set M of $f(d)t^d$ elements such that if there is a solution with t sets, then there is such a solution inside M.

Marking

Another interpretation:

We can mark a set M of $f(|V(H)|)k^{|V(H)|}$ vertices such that the following holds. If Z is any set of at most k vertices and there is a copy of H disjoint from Z, then there is such a copy inside M.

$$\Downarrow$$

In the H-Packing problem, we can mark a set M of $f(d)k^{|V(H)|}$ vertices (where $k = t \cdot |V(H)|$) such that if there is solution, then there is a solution inside M.

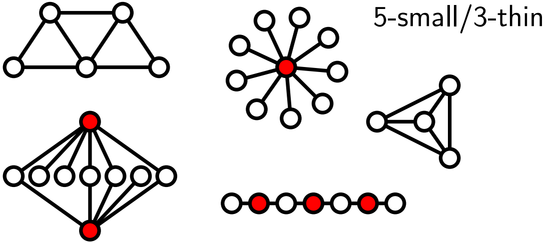
Bottom line:
We need marking procedures of this form for packing problems.

Kernels for $\mathcal{F}$-Packing

Definition

A graph is a-small/b-thin if every connected component

- has at most a vertices, or
- is a bipartite graph whose smallest size has at most b vertices.



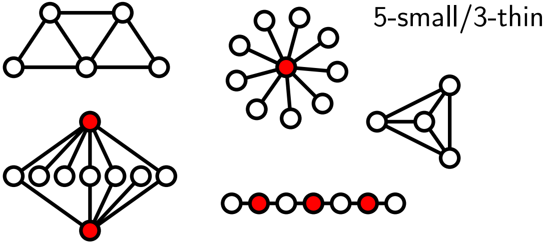
$\mathcal{F}$ is small/thin if $\exists a, b \geq 0$ such that every $H \in \mathcal{F}$ is a-small/b-thin.

Kernels for $\mathcal{F}$-Packing

Definition

A graph is a-small/b-thin if every connected component

- has at most a vertices, or
- is a bipartite graph whose smallest size has at most b vertices.



Theorem

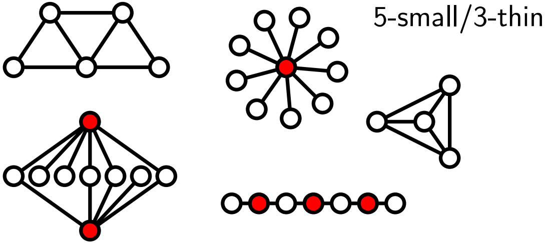
$\mathcal{F}$-Packing admits a many-one polynomial kernel if $\mathcal{F}$ is small/thin, and otherwise does not have a polynomial kernel (unless NP $\subseteq$ coNP/poly).

Kernels for $\mathcal{F}$-Packing

Definition

A graph is a-small/b-thin if every connected component

- has at most a vertices, or
- is a bipartite graph whose smallest size has at most b vertices.



Theorem

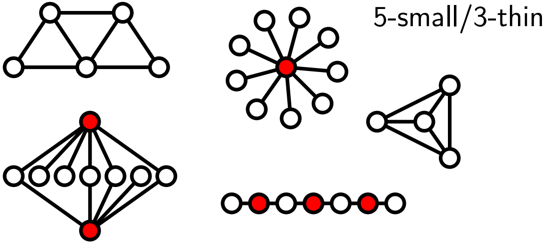
$\mathcal{F}$-Packing admits a polynomial Turing kernel if $\mathcal{F}$ is small/thin, and otherwise W[1]-hard, WK[1]-hard, or Long Path-hard.

Kernels for $\mathcal{F}$-PACKING

Definition

A graph is a-small/b-thin if every connected component

- has at most a vertices, or
- is a bipartite graph whose smallest size has at most b vertices.



Turing kernels do not buy us more power for $\mathcal{F}$-PACKING!

Ingredients for $\mathcal{F}$-Packing kernelization dichotomy

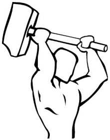
Classification

Small/thin graph classes characterize the easy cases.

Ingredients for $\mathcal{F}$-Packing kernelization dichotomy

Classification

Small/thin graph classes characterize the easy cases.



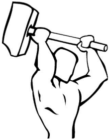
Algorithms

Marking procedure based on the Sunflower lemma for small components and on problem-specific arguments for thin bipartite components.

Ingredients for $\mathcal{F}$-PACKING kernelization dichotomy

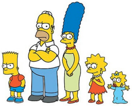
Classification

Small/thin graph classes characterize the easy cases.



Algorithms

Marking procedure based on the Sunflower lemma for small components and on problem-specific arguments for thin bipartite components.



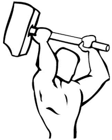
Hard families

Kernelization lower bound for each hard family by polynomial-parameter transformations from UNIFORM EXACT SET COVER.

Ingredients for $\mathcal{F}$-PACKING kernelization dichotomy

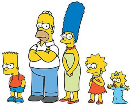
Classification

Small/thin graph classes characterize the easy cases.



Algorithms

Marking procedure based on the Sunflower lemma for small components and on problem-specific arguments for thin bipartite components.



Hard families

Kernelization lower bound for each hard family by polynomial-parameter transformations from UNIFORM EXACT SET COVER.

Ramsey arguments

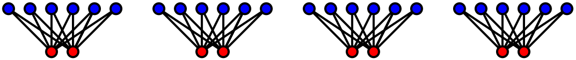
Hereditary $\mathcal{F}$ that is not small/thin contains one of the hard families.

Packing thin bicliques

A special case of the kernelization result:

Theorem

$K_{x,y}$-PACKING admits a a polynomial kernel for every fixed x (y is part of the input).



We need a marking procedure:

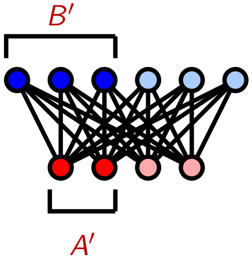
We can mark a set M of $k^{O(x)}$ vertices such that the following holds. If Z is any set of at most k vertices and there is a copy of $K_{x,y}$ disjoint from Z, then there is a copy in $M \setminus Z$.

Marking procedure for thin bicliques

We prove a more technical statement:

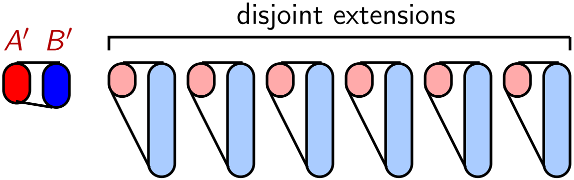
For every (A', B'), we can mark a set M of $k^{O(x)}$ vertices such that the following holds. If Z is any set of at most k vertices and there is a copy of $K_{x,y}$ *extending* (A', B') and disjoint from Z, then there is a copy of $K_{x,y}$ in $M \setminus Z$. [Not necessarily extending (A', B')!].

A copy (A, B) of $K_{x,y}$ **extends** (A', B') if $A' \subseteq A$ and $B' \subseteq B$.



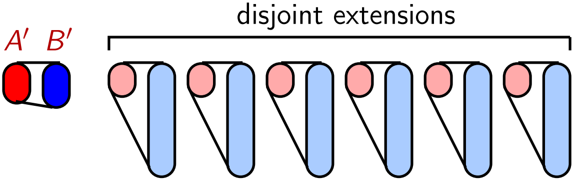
Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.



Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.

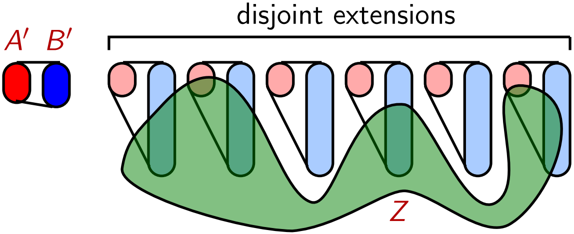


Main step:

- If there are $k + 1$ copies: done.

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.

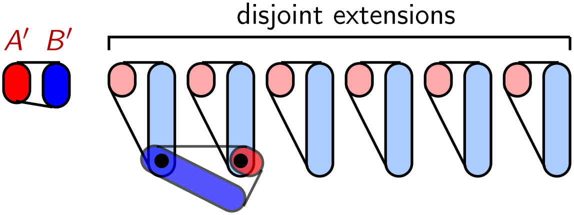


Main step:

- If there are $k+1$ copies: done.

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.

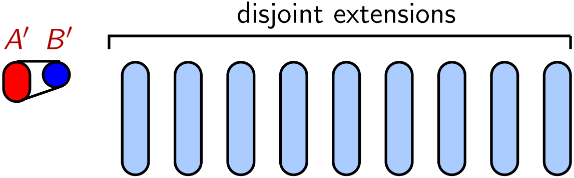


Main step:

- If there are $k + 1$ copies: done.
- If there are at most k copies: branch on including into A' or B' each of the at most $k(x + y)$ vertices of the copies.

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.

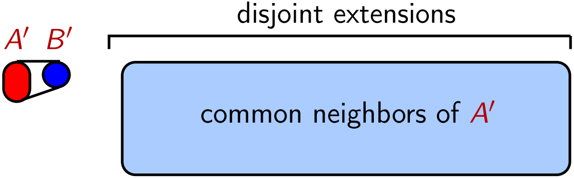


Corner case 1: $|A'| = x$

The extensions are just common neighbors of A'.

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.



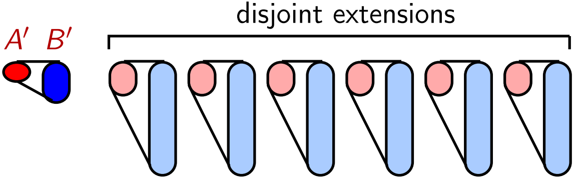
Corner case 1: $|A'| = x$

The extensions are just common neighbors of A'.

Mark $k + y$ common neighbors of A' (or all of them, if they are fewer).

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.

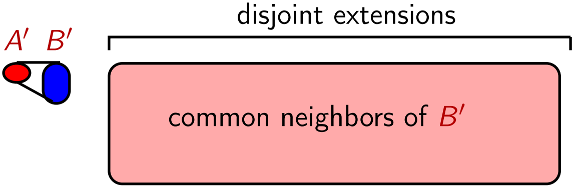


Corner case 2: $|A'| < x$, $|B'| = x$

The extensions are just common neighbors of B'.

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.



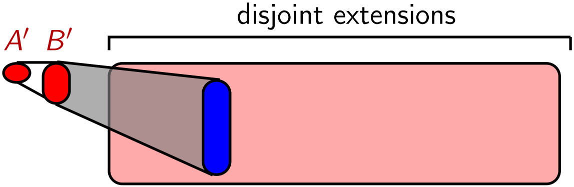
Corner case 2: $|A'| < x$, $|B'| = x$

The extensions are just common neighbors of B'.

- If B' has less than $k + y$ common neighbors, then branch on including one of them into A'.

Marking procedure for thin bicliques

Greedily find copies of $K_{x,y}$ extending (A', B') that meet only in $A' \cup B'$.



Corner case 2: $|A'| < x$, $|B'| = x$

The extensions are just common neighbors of B'.

- If B' has less than $k + y$ common neighbors, then branch on including one of them into A'.
- If B' has at least $k + y$ common neighbors, then mark $k + y$ of them and we are done: B' **and any** y **common neighbors of** B' **form a** $K_{x,y}$**!**

Packing thin bicliques

The recursive marking procedure branches into at most $2k(x+y) \leq 2k^2$ directions and the recursion depth is at most $2x$
$\Rightarrow$ at most $k^{O(x)}$ vertices are marked.

We can mark a set M of $k^{O(x)}$ vertices such that the following holds. If Z is any set of at most k vertices and there is a copy of $K_{x,y}$ disjoint from Z, then there is a copy in $M \setminus Z$.

Packing thin bicliques

The recursive marking procedure branches into at most $2k(x + y) \leq 2k^2$ directions and the recursion depth is at most $2x$ $\Rightarrow$ at most $k^{O(x)}$ vertices are marked.

We can mark a set M of $k^{O(x)}$ vertices such that the following holds. If Z is any set of at most k vertices and there is a copy of $K_{x,y}$ disjoint from Z, then there is a copy in $M \setminus Z$.

Theorem

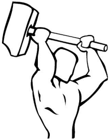
$K_{x,y}$-Packing admits a a polynomial kernel for every fixed x (y is part of the input).

The marking procedure can be extended to arbitrary thin bipartite graphs, but it is much more technical.

Ingredients for $\mathcal{F}$-Packing kernelization dichotomy

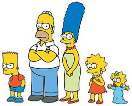
Classification

Small/thin graph classes characterize the easy cases.



Algorithms

Marking procedure based on the Sunflower lemma for small components and on problem-specific arguments for thin bipartite components.



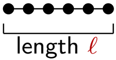
Hard families

Kernelization lower bound for each hard family by polynomial-parameter transformations from Uniform Exact Set Cover.

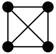
Ramsey arguments

Hereditary $\mathcal{F}$ that is not small/thin contains one of the hard families.

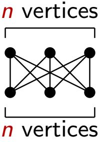
Hard families



Path(ℓ)
($\ell = 5$)



Clique(n)
($n = 4$)



Biclique(n)
($n = 3$)

2-broom(s, n)

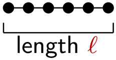
Fountain(s, n)

LongFountain(s, t, n)

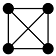
SubDivStar(n)

OperaHouse(s, n)

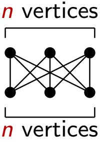
Hard families



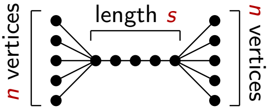
Path(ℓ)
($\ell = 5$)



Clique(n)
($n = 4$)



Biclique(n)
($n = 3$)



2-broom(s, n)
($s = 4, n = 5$)

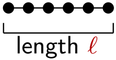
Fountain(s, n)

LongFountain(s, t, n)

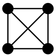
SubDivStar(n)

OperaHouse(s, n)

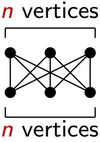
Hard families



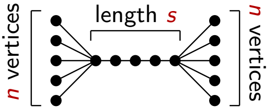
Path(ℓ)
($\ell = 5$)



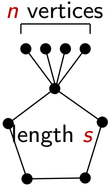
Clique(n)
($n = 4$)



Biclique(n)
($n = 3$)



2-broom(s, n)
($s = 4$, $n = 5$)



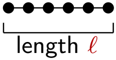
Fountain(s, n)
($s = 5$, $n = 4$)

LongFountain(s, t, n)

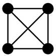
SubDivStar(n)

OperaHouse(s, n)

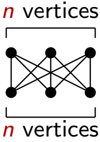
Hard families



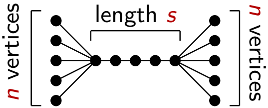
Path(ℓ)
($\ell = 5$)



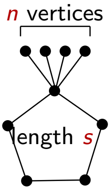
Clique(n)
($n = 4$)



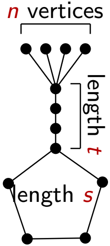
Biclique(n)
($n = 3$)



2-broom(s, n)
($s = 4$, $n = 5$)



Fountain(s, n)
($s = 5$, $n = 4$)

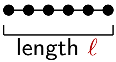


LongFountain(s, t, n)
($s = 5$, $t = 3$, $n = 4$)

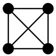
SubDivStar(n)

OperaHouse(s, n)

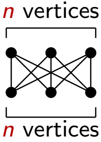
Hard families



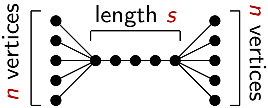
Path(ℓ)
($\ell = 5$)



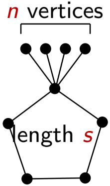
Clique(n)
($n = 4$)



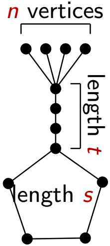
Biclique(n)
($n = 3$)



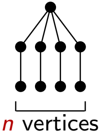
2-broom(s, n)
($s = 4$, $n = 5$)



Fountain(s, n)
($s = 5$, $n = 4$)



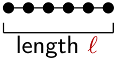
LongFountain(s, t, n)
($s = 5$, $t = 3$, $n = 4$)



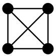
SubDivStar(n)
($n = 4$)

OperaHouse(s, n)

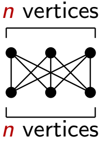
Hard families



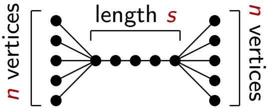
Path(ℓ)
($\ell = 5$)



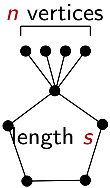
Clique(n)
($n = 4$)



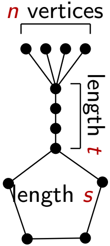
Biclique(n)
($n = 3$)



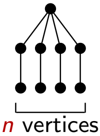
2-broom(s, n)
($s = 4, n = 5$)



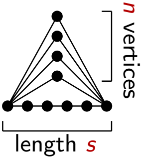
Fountain(s, n)
($s = 5, n = 4$)



LongFountain(s, t, n)
($s = 5, t = 3, n = 4$)

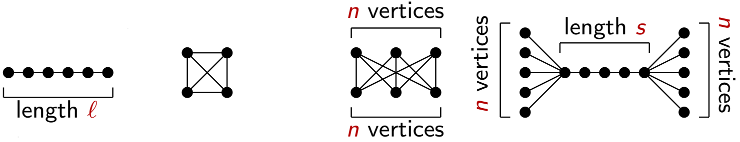


SubDivStar(n)
($n = 4$)

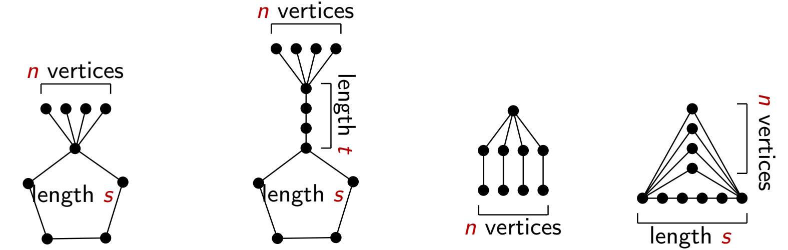


OperaHouse(s, n)
($s = 5, n = 4$)

Hard families



Path(ℓ) ($\ell = 5$) Clique(n) ($n = 4$) Biclique(n) ($n = 3$) 2-broom(s, n) ($s = 4, n = 5$)



Fountain(s, n) ($s = 5, n = 4$) LongFountain(s, t, n) ($s = 5, t = 3, n = 4$) SubDivStar(n) ($n = 4$) OperaHouse(s, n) ($s = 5, n = 4$)

We show e.g. that if $\{\text{LongFountain}(5, 2, n) \mid n \geq 1\} \subseteq \mathcal{F}$, then $\mathcal{F}$-Packing is WK[1]-hard.

Hard families

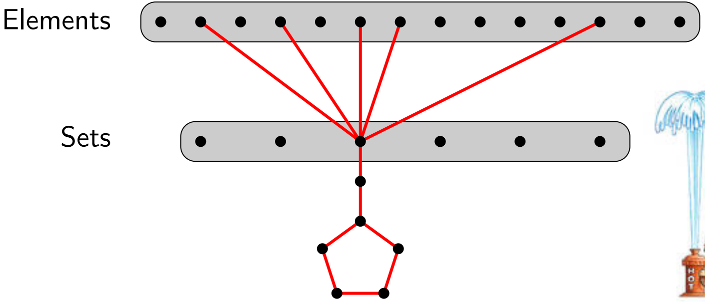
Theorem

$\mathcal{F}$-PACKING is WK[1]-hard if one of the following holds:

- $\{\text{SubDivStar}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{Fountain}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$,
- $\{\text{LongFountain}(s, t, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some integer $t \geq 1$ and odd integer $s \geq 3$,
- $\{\text{2-broom}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$, or
- $\{\text{OperaHouse}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$.

Reducion from
Uniform Exact Set Cover

Version of Set Cover where every set has the same size n/k.



Reduction to $\mathcal{F}$-Packing if $\{\mathrm{LongFountain}(5, 2, n) \mid n \geq 1\} \subseteq \mathcal{F}$.

Ramsey arguments

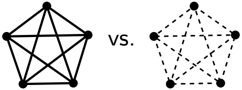
Theorem

If a hereditary class $\mathcal{F}$ is not small/thin, then at least one of the following holds:

- $\{\text{Path}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{Clique}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{Biclique}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{SubdivStar}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{Fountain}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$,
- $\{\text{LongFountain}(s, t, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some integer $t \geq 1$ and odd integer $s \geq 3$,
- $\{\text{2-broom}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$, or
- $\{\text{OperaHouse}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$.

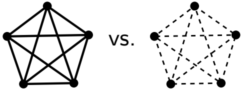
Ramsey-type arguments

- A large graph has a large clique or independent set.

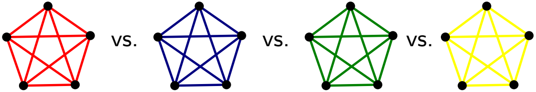


Ramsey-type arguments

- A large graph has a large clique or independent set.

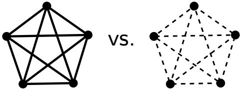


- A large c-edge-colored clique has a large monochromatic clique.

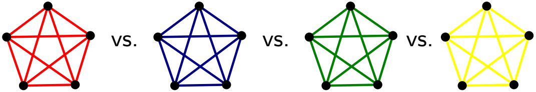


Ramsey-type arguments

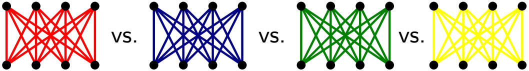
- A large graph has a large clique or independent set.



- A large c-edge-colored clique has a large monochromatic clique.

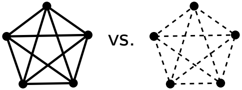


- A large c-edge-colored biclique has a large monochromatic biclique.

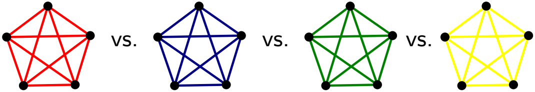


Ramsey-type arguments

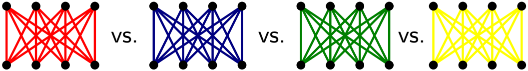
- A large graph has a large clique or independent set.



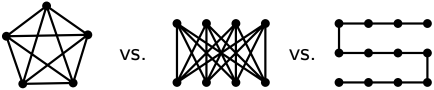
- A large *c*-edge-colored clique has a large monochromatic clique.



- A large *c*-edge-colored biclique has a large monochromatic biclique.

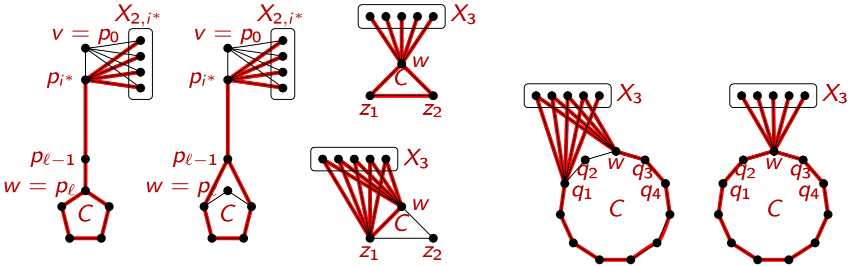


- If a graph has a long path, then it has a large induced path, a clique, or an induced biclique. [Galvin, Rival, Sands 1982]



Ramsey arguments

Need to show: if a connected nonbipartite graph is large, then it contains a large bad guy.



Case 1.a Case 1.b Case 2.a Case 2.b

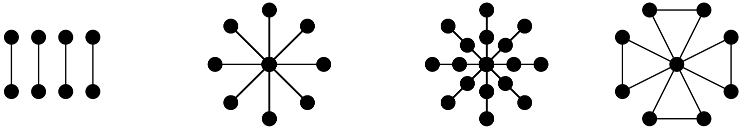
Observation: if there is no long induced path, then a large component has to contain a vertex of large degree.

Finding subgraphs in polynomial time

Subgraph Isomorphism	
Input:	two graphs H and G.
Task:	decide if G has a subgraph isomorphic to H.

Some classes for which $\mathcal{F}$-Subgraph Isomorphism is polynomial-time solvable:

- $\mathcal{F}$ is the class of all matchings
- $\mathcal{F}$ is the class of all stars
- $\mathcal{F}$ is the class of all stars, each edge subdivided once
- $\mathcal{F}$ is the class of all windmills

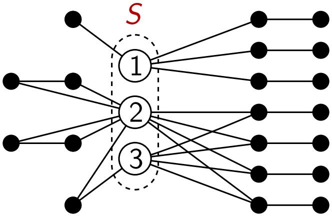


matching star subdivided star windmill

Finding subgraphs

Definition

Class $\mathcal{F}$ is **matching splittable** if there is a constant c such that every $H \in \mathcal{F}$ has a set S of at most c vertices such that every component of $H - S$ has size at most 2.



Theorem

Let $\mathcal{F}$ be a hereditary class of graphs. If $\mathcal{F}$ is matching splittable, then $\mathcal{F}$-Subgraph Isomorphism is randomized polynomial-time solvable and NP-hard otherwise.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism polynomial-time dichotomy

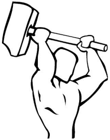
Classification

Matching splittable graph families characterize the easy cases.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism polynomial-time dichotomy

Classification

Matching splittable graph families characterize the easy cases.



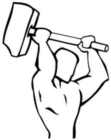
Algorithms

Algorithm by guessing a few vertices + reduction to colored matching.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism polynomial-time dichotomy

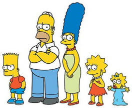
Classification

Matching splittable graph families characterize the easy cases.



Algorithms

Algorithm by guessing a few vertices + reduction to colored matching.



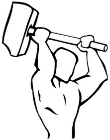
Hard families

Finding cliques, bicliques, $n \cdot P_3$, and $n \cdot K_3$ are all NP-hard.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism polynomial-time dichotomy

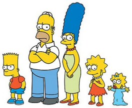
Classification

Matching splittable graph families characterize the easy cases.



Algorithms

Algorithm by guessing a few vertices + reduction to colored matching.



Hard families

Finding cliques, bicliques, $n \cdot P_3$, and $n \cdot K_3$ are all NP-hard.

Ramsey arguments

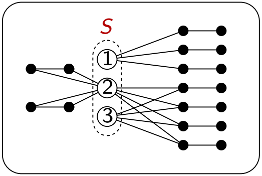
Hereditary $\mathcal{F}$ that is not matching splittable contains either all cliques, bicliques, $n \cdot P_3$, or $n \cdot K_3$.

Finding subgraphs (algorithm)

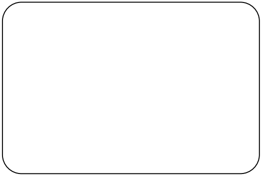
Theorem

If hereditary class $\mathcal{F}$ is matching splittable, then $\mathcal{F}$-Subgraph Isomorphism is randomized polynomial-time solvable.

H



G

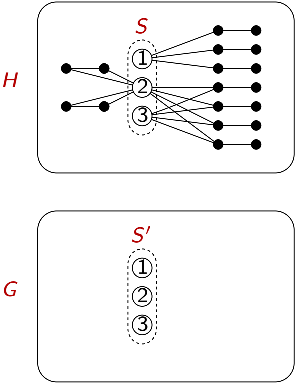


Finding subgraphs (algorithm)

Theorem

If hereditary class $\mathcal{F}$ is matching splittable, then $\mathcal{F}$-SUBGRAPH ISOMORPHISM is randomized polynomial-time solvable.

- Guess the image S' of S in G.

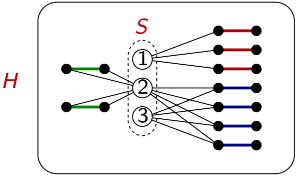


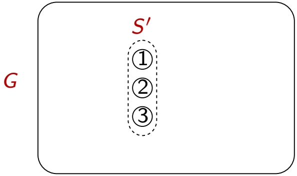
Finding subgraphs (algorithm)

Theorem

If hereditary class $\mathcal{F}$ is matching splittable, then $\mathcal{F}$-Subgraph Isomorphism is randomized polynomial-time solvable.

- Guess the image S' of S in G.
- Classify the edges of $H - S$ according to their neighborhoods in S (at most 2^{2c} colors).



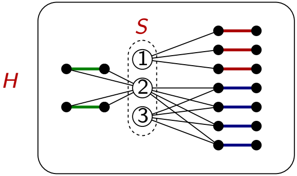


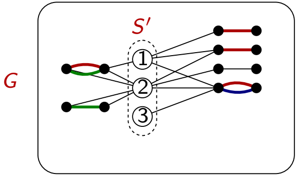
Finding subgraphs (algorithm)

Theorem

If hereditary class $\mathcal{F}$ is matching splittable, then $\mathcal{F}$-Subgraph Isomorphism is randomized polynomial-time solvable.

- Guess the image S' of S in G.
- Classify the edges of $H - S$ according to their neighborhoods in S (at most 2^{2c} colors).
- Classify the edges of $G - S'$ according to which edge of $H - S$ can be mapped into it (use parallel edges if needed).
- Task is to find a matching in $G - S'$ with a certain number of edges of each color.





Finding subgraphs (algorithm)

Theorem [Mulmuley, Vazirani, Vazirani 1987]

There is a randomized polynomial-time algorithm that, given a graph G with red and blue edges and integer k, decides if there is a perfect matching with exactly k red edges.

More generally:

Theorem

Given a graph G with edges colored with c colors and c integers k_1, $\ldots$, k_c, we can decide in randomized time $n^{O(c)}$ if there is a matching with exactly k_i edges of color i.

This is precisely what we need to complete the algorithm for $\mathcal{F}$-Subgraph Isomorphism for matching splittable $\mathcal{F}$.

Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{F}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{F}$ contains every clique.
- $\mathcal{F}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot P_3$ (where P_3 is the path on 3 vertices).

In each case, $\mathcal{F}$-Subgraph Isomorphism is NP-hard (recall that P_3-Packing and K_3-Packing are NP-hard).

Finding subgraphs (hardness proof)

Definition

Class $\mathcal{F}$ is **matching splittable** if there is a constant c such that every $H \in \mathcal{F}$ has a set S of at most c vertices such that every component of $H - S$ has size at most 2.

Equivalently: in every $H \in \mathcal{F}$, we can cover every 3-vertex connected set (i.e., every K_3 and P_3) by c vertices.

Observation: either

- there are r vertex-disjoint copies of K_3, or
- there are r vertex-disjoint copies of P_3, or
- we can cover every K_3 and every P_3 by $6r$ vertices.

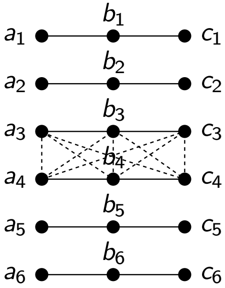
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{F}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{F}$ contains every clique.
- $\mathcal{F}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.



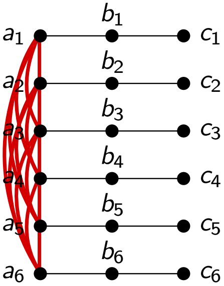
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{F}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{F}$ contains every clique.
- $\mathcal{F}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.

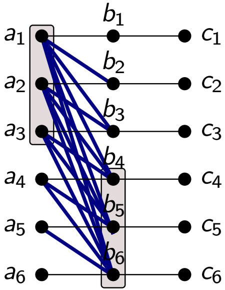


Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{F}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{F}$ contains every clique.
- $\mathcal{F}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot P_3$.



- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.

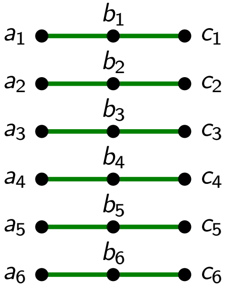
Finding subgraphs (hardness proof)

Lemma

Let $\mathcal{F}$ be a hereditary class of graphs that is not matching splittable. Then at least one of the following is true.

- $\mathcal{F}$ contains every clique.
- $\mathcal{F}$ contains every biclique.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot K_3$.
- For every $n \geq 1$, $\mathcal{F}$ contains $n \cdot P_3$.

- Consider many vertex-disjoint P_3's.
- For every $i < j$, there are 2^9 possibilities between $\{a_i, b_i, c_i\}$ and $\{a_j, b_j, c_j\}$.
- There is a homogeneous set of many P_3's with respect to these 2^9 possibilities.
- In each of the 2^9 cases, we find many disjoint P_3's, a clique, or a biclique.



Finding subgraphs

What did we learn from the polynomial-time dichotomy?

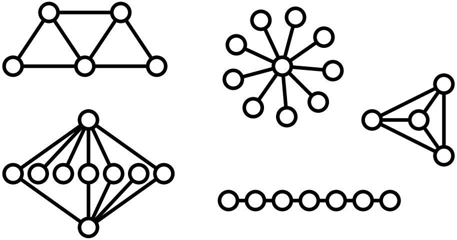
Guessing the locations of a few vertices can be really important for finding subgraphs!

- Turing kernels can guess the locations of a few vertices and produce a polynomial kernel for each guess.
- But this can be a real problem for many-one kernels.

As we shall see, this leads to a difference in power between the two types of kernelizations.

Universal vertices

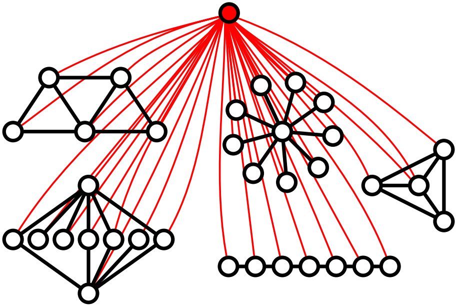
If every component of H is small/thin, then we can kernelize using the marking procedure used for the packing problem.



With Turing kernelization, we can do more: we have a Turing kernel even if we attach a constant number of universal vertices.

Universal vertices

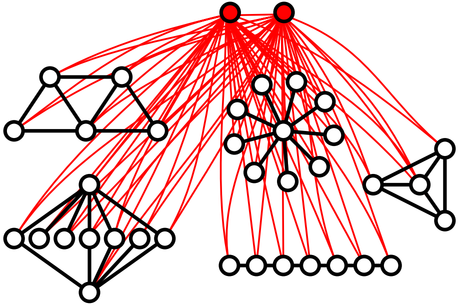
If every component of H is small/thin, then we can kernelize using the marking procedure used for the packing problem.



With Turing kernelization, we can do more: we have a Turing kernel even if we attach a constant number of universal vertices.

Universal vertices

If every component of H is small/thin, then we can kernelize using the marking procedure used for the packing problem.



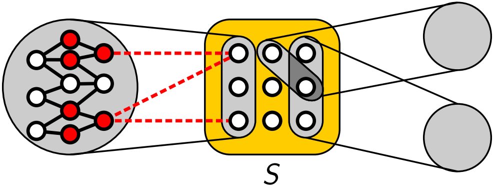
With Turing kernelization, we can do more: we have a Turing kernel even if we attach a constant number of universal vertices.

Splittable graphs

Definition

A graph H is (a, b, c, d)-**splittable** if it has a set S of at most c vertices such that

- if $H - S$ is a-small/b-thin, and
- each component C of $H - S$ has at most d vertices whose closed neighborhood in $G[C]$ is not universal to $N_H(C) \cap S$.



$\mathcal{F}$ is splittable if $\exists a, b, c, d$ such that every $F \in \mathcal{F}$ is (a, b, c, d)-splittable.

Splittable graphs

Definition

A graph H is (a, b, c, d)-**splittable** if it has a set S of at most c vertices such that

- if $H - S$ is a-small/b-thin, and
- each component C of $H - S$ has at most d vertices whose closed neighborhood in $G[C]$ is not universal to $N_H(C) \cap S$.

Theorem

If $\mathcal{F}$ is a splittable hereditary class, then $\mathcal{F}$-Subgraph Isomorphism admits a polynomial Turing kernel, otherwise it is W[1]-hard, WK[1]-hard, or Long Path-hard.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism Turing kernelization dichotomy

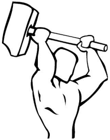
Classification

Splittable graph families characterize the easy cases.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism Turing kernelization dichotomy

Classification

Splittable graph families characterize the easy cases.



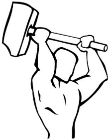
Algorithms

Algorithm by guessing a few vertices + marking procedure for small/thin components.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism Turing kernelization dichotomy

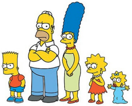
Classification

Splittable graph families characterize the easy cases.



Algorithms

Algorithm by guessing a few vertices + marking procedure for small/thin components.

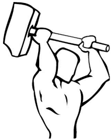


Hard families

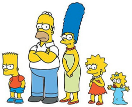
Hard families coming from the packing problem + two new hard families specific for subgraph isomorphism.

Ingredients for $\mathcal{F}$-Subgraph Isomorphism Turing kernelization dichotomy

Classification
Splittable graph families characterize the easy cases.



Algorithms
Algorithm by guessing a few vertices + marking procedure for small/thin components.

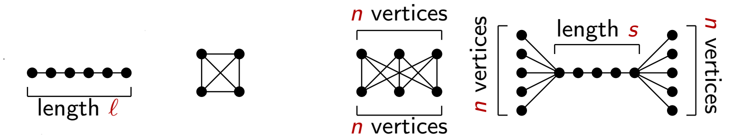


Hard families
Hard families coming from the packing problem + two new hard families specific for subgraph isomorphism.

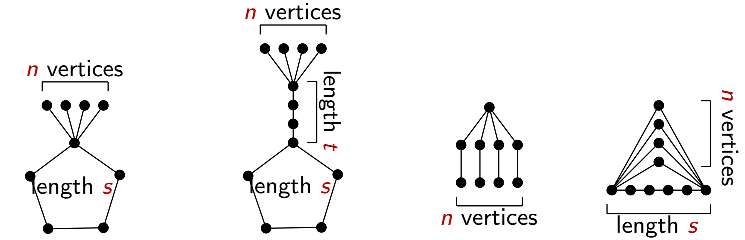
Ramsey arguments
Hereditary $\mathcal{F}$ that is not splittable contains at least one of the hard families.

Hard families

Hardness results coming from the hardness of packing:



Path(ℓ) ($\ell = 5$) | Clique(n) ($n = 4$) | Biclique(n) ($n = 3$) | 2-broom(s, n) ($s = 4, n = 5$)

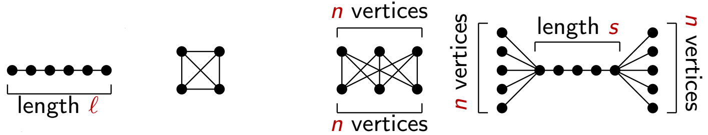


Fountain(s, n) ($s = 5, n = 4$) | LongFountain(s, t, n) ($s = 5, t = 3, n = 4$) | SubDivStar(n) ($n = 4$) | OperaHouse(s, n) ($s = 5, n = 4$)

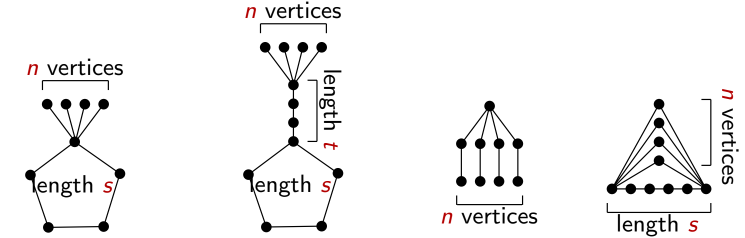
If $\{\text{LongFountain}(5, 2, n) \mid n \geq 1\} \subseteq \mathcal{F}$, then $\mathcal{F}$-Packing is WK[1]-hard.

Hard families

Hardness results coming from the hardness of packing:



Path(ℓ) ($\ell = 5$) Clique(n) ($n = 4$) Biclique(n) ($n = 3$) 2-broom(s, n) ($s = 4, n = 5$)

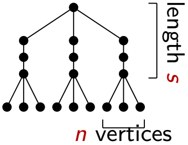


Fountain(s, n) ($s = 5, n = 4$) LongFountain(s, t, n) ($s = 5, t = 3, n = 4$) SubDivStar(n) ($n = 4$) OperaHouse(s, n) ($s = 5, n = 4$)

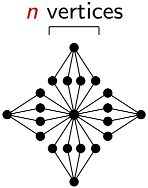
If $\{n \cdot \text{LongFountain}(5, 2, n) \mid n \geq 1\} \subseteq \mathcal{F}$, then $\mathcal{F}$-Subgraph Isomorphism is WK[1]-hard.

Hard families

Two new types of hard families:



SubDivTree(s, n)
($s = 3, n = 3$)



DiamondFan(n)
($n = 4$)

We prove that $\mathcal{F}$-Subgraph Isomorphism is WK[1]-hard if

- $\{\text{DiamondFan}(n) \mid n \geq 1\} \subseteq \mathcal{F}$ or
- $\{\text{SubDivTree}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some $s \geq 1$.

Ramsey arguments

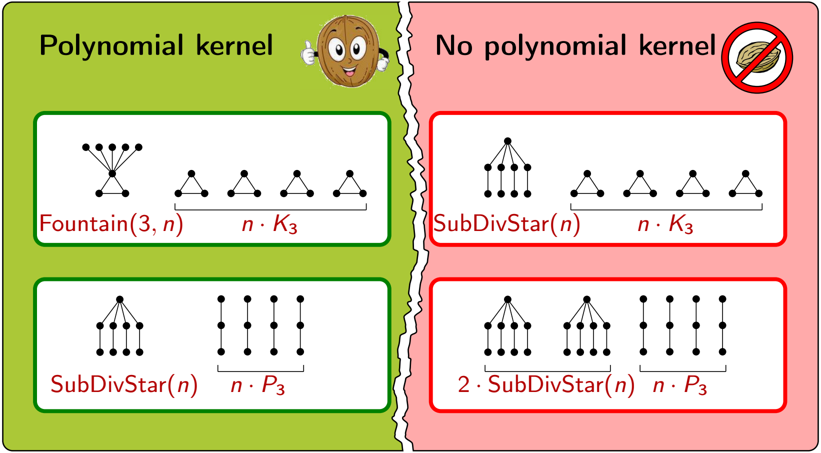
Theorem

If a hereditary class $\mathcal{F}$ is not splittable, then at least one of the following holds:

- $\{\text{Path}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{Clique}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{\text{Biclique}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{n \cdot \text{SubDivStar}(n) \mid n \geq 1\} \subseteq \mathcal{F}$,
- $\{n \cdot \text{Fountain}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$,
- $\{n \cdot \text{LongFountain}(s, t, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some integer $t \geq 1$ and odd integer $s \geq 3$,
- $\{n \cdot \text{2-broom}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$,
- $\{n \cdot \text{OperaHouse}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some odd integer $s \geq 3$,
- $\{\text{SubDivTree}(s, n) \mid n \geq 1\} \subseteq \mathcal{F}$ for some integer $s \geq 1$, or
- $\{\text{DiamondFan}(n) \mid n \geq 1\} \subseteq \mathcal{F}$.

Many-one kernels for SUBGRAPH ISOMORPHISM

The landscape of many-one kernels is very confusing.



Summary

- **Goal:** dichotomies for PACKING and SUBGRAPH ISOMORPHISM from the viewpoints of
  - polynomial-time algorithms,
  - many-one kernels,
  - and Turing kernels.
- The project was doable, except for many-one kernelization for SUBGRAPH ISOMORPHISM
- For PACKING, Turing kernels do not give us more power than many-one kernels.
- Guessing a few vertices seems to be a very basic step for SUBGRAPH ISOMORPHISM.
- Why was not the polynomial-time dichotomy for SUBGRAPH ISOMORPHISM known earlier?