Some Contributions to Parameterized Complexity

Habilitation à Diriger des Recherches (HDR)

Montpellier, June 25, 2018

Ignasi Sau

CNRS, LIRMM, Université de Montpellier

***Reviewers*:**	Michael R. Fellows	-	University of Bergen
	Rolf Niedermeier	-	Technische Universität Berlin
	Fedor V. Fomin	-	University of Bergen
***Examinators*:**	Jean-Claude Bermond	-	CNRS, U. de Nice-Sophia Antipolis
	Marc Noy	-	Univ. Politècnica de Catalunya
	Dimitrios M. Thilikos	-	CNRS, Université de Montpellier
	Gilles Trombettoni	-	Université de Montpellier

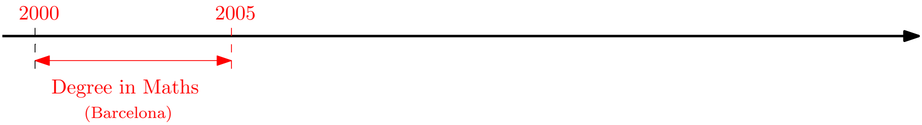
Outline of the talk

1. Introduction
   - Career path
   - Scientific context: parameterized complexity
   - A relevant parameter: treewidth
2. Some of my contributions (related to treewidth)
   - The number of graphs of bounded treewidth
   - Linear kernels on sparse graphs
   - Fast FPT algorithms parameterized by treewidth
3. Conclusions

Next section is...

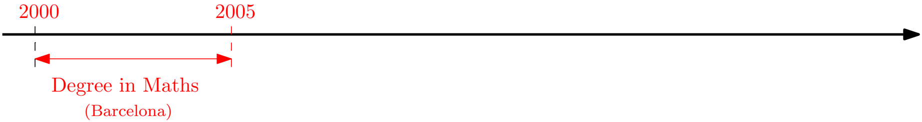
My (scientific) life in one slide

My (scientific) life in one slide



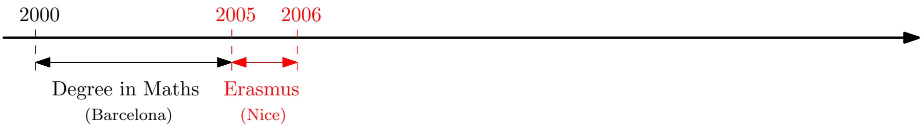
- Maths + Telecom at Universitat Politècnica de Catalunya (UPC).

My (scientific) life in one slide



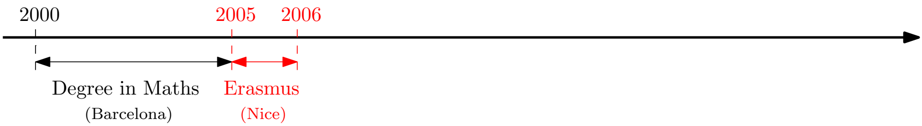
- Maths + Telecom at Universitat Politècnica de Catalunya (UPC).

My (scientific) life in one slide

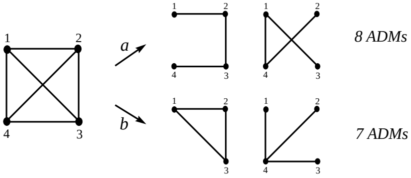


- My first contact with France.
- **Topic**: traffic grooming in optical networks.

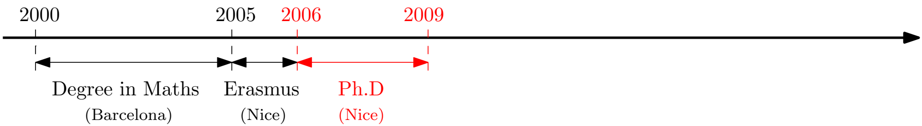
My (scientific) life in one slide



- My first contact with France.
- **Topic**: traffic grooming in optical networks.

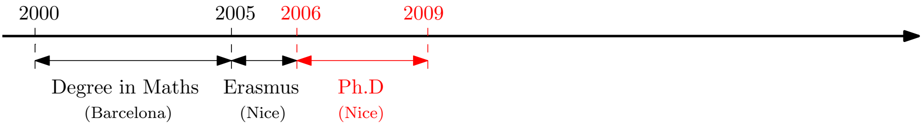


My (scientific) life in one slide



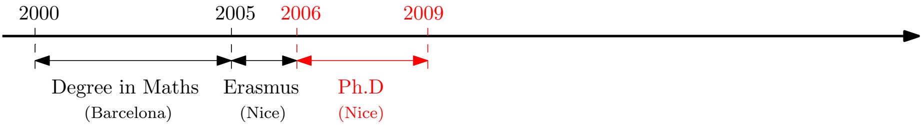
- Advisors: X. Muñoz (Barcelona) + D. Coudert, J.-C. Bermond (Sophia).
- **Topic**: optimization in graphs under degree constraints.

My (scientific) life in one slide

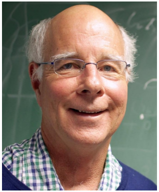


- I started collaborating with Dimitrios M. Thilikos.
- I converted to parameterized complexity.

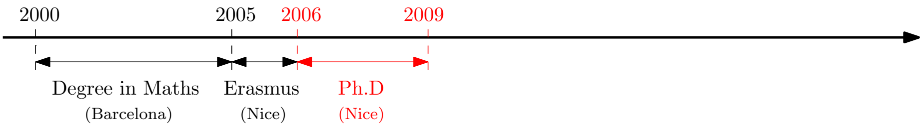
My (scientific) life in one slide



- I started collaborating with Dimitrios M. Thilikos.
- I converted to parameterized complexity.

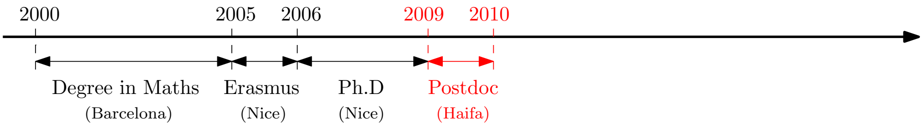


My (scientific) life in one slide

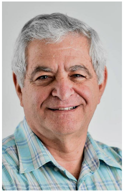


- I started collaborating with Dimitrios M. Thilikos.
- I converted to parameterized complexity.

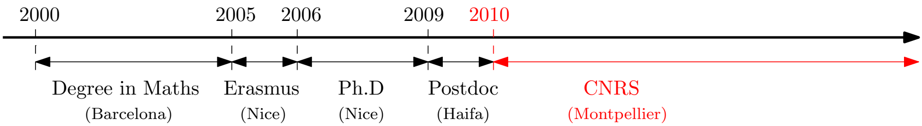
My (scientific) life in one slide



- Postdoc at the Computer Science Department of the Technion.
- With Shmuel Zaks and Mordechai Shalom.

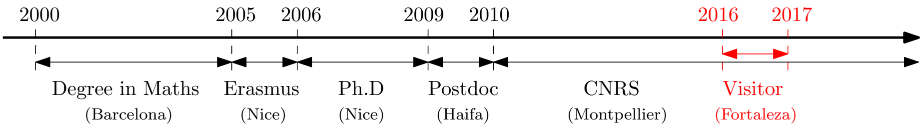


My (scientific) life in one slide



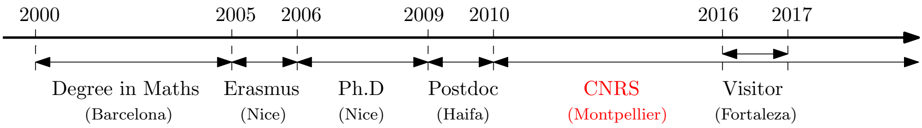
- Since October 2010, I joined the CNRS at LIRMM, Montpellier.
- AIGCo group: Algorithmes, Graphes et Combinatoire.

My (scientific) life in one slide



- Visiting professor at Universidade Federal do Ceará, Fortaleza, Brazil.
- ParGO group: Paralelismo, Grafos e Otimização combinatòria.

My (scientific) life in one slide



- Since August 2017, back to Montpellier.
- AIGCo group: Algorithmes, Graphes et Combinatoire.

Supervised students

- 03/**2012**-08/**2012** Valentin Garnero (internship M2)
  Polynomial kernels for variants of domination problems on planar graphs
- 02/**2013**-07/**2013** Julien Baste (internship M2)
  The role of planarity in connectivity problems parameterized by treewidth
- 02/**2014**-07/**2014** Henri Perret du Cray (internship M2)
  FPT algorithms and kernels on graphs without induced subgraphs

Supervised students

- 03/**2012**-08/**2012** Valentin Garnero (internship M2)
  Polynomial kernels for variants of domination problems on planar graphs
- 02/**2013**-07/**2013** Julien Baste (internship M2)
  The role of planarity in connectivity problems parameterized by treewidth
- 02/**2014**-07/**2014** Henri Perret du Cray (internship M2)
  FPT algorithms and kernels on graphs without induced subgraphs

- 10/**2012**-07/**2016** Valentin Garnero (**Ph.D**, with Christophe Paul)
  (Méta)-noyaux constructifs et linéaires dans les graphes peu denses
- 09/**2014**-09/**2017** Julien Baste (**Ph.D**, with Dimitrios M. Thilikos)
  Treewidth: algorithmic, combinatorial and practical aspects

Supervised students

- 03/**2012**-08/**2012** Valentin Garnero (internship M2)
  Polynomial kernels for variants of domination problems on planar graphs
- 02/**2013**-07/**2013** Julien Baste (internship M2)
  The role of planarity in connectivity problems parameterized by treewidth
- 02/**2014**-07/**2014** Henri Perret du Cray (internship M2)
  FPT algorithms and kernels on graphs without induced subgraphs

- 10/**2012**-07/**2016** Valentin Garnero (**Ph.D**, with Christophe Paul)
  (Méta)-noyaux constructifs et linéaires dans les graphes peu denses
- 09/**2014**-09/**2017** Julien Baste (**Ph.D**, with Dimitrios M. Thilikos)
  Treewidth: algorithmic, combinatorial and practical aspects

- 09/**2018**-08/**2019** Raul Wayne (Ph.D internship, Brazil)
  Fixed-parameter tractability of the Directed Grid Theorem
- 09/**2018**-03/**2019** Guilherme Gomes (Ph.D internship, Brazil)
  Cliques, bicliques and colorings

Next subsection is...

Some history of complexity: NP-completeness

- Cook-Levin Theorem (1971): the SAT problem is NP-complete.
- Karp (1972): list of 21 *important* NP-complete problems.
- Nowadays, literally thousands of problems are known to be NP-hard: unless $\mathsf{P} = \mathsf{NP}$, they cannot be solved in polynomial time.

Some history of complexity: NP-completeness

- Cook-Levin Theorem (1971): the SAT problem is NP-complete.
- Karp (1972): list of 21 *important* NP-complete problems.
- Nowadays, literally thousands of problems are known to be NP-hard: unless $\mathrm{P} = \mathrm{NP}$, they cannot be solved in polynomial time.
- But what does it mean for a problem to be NP-hard?

  No algorithm solves all instances optimally in polynomial time.

Are all instances really hard to solve?

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

Are all instances really hard to solve?

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

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

Are all instances really hard to solve?

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

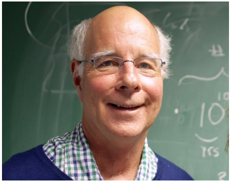
- VLSI design: the number of circuit layers is usually ≤ 10.
- Computational biology: Real instances of DNA chain reconstruction usually have treewidth ≤ 11.
- Robotics: Number of degrees of freedom in motion planning problems ≤ 10.
- Compilers: Checking compatibility of type declarations is hard, but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

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

The area of parameterized complexity

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

This theory started in the late 80's, by Downey and Fellows:



Today, it is a well-established area with hundreds of articles published every year in the most prestigious TCS journals and conferences.

Parameterized problems

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet.

For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

Parameterized problems

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet.

For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

- k-Vertex Cover: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \leq k$, containing at least an endpoint of every edge?
- k-Independent Set: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \geq k$, of pairwise non-adjacent vertices?
- Vertex k-Coloring: Can the vertices of a graph be colored with $\leq k$ colors, so that any two adjacent vertices get different colors?

Parameterized problems

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite alphabet.

For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.

- k-Vertex Cover: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \leq k$, containing at least an endpoint of every edge?
- k-Independent Set: Does a graph G contain a set $S \subseteq V(G)$, with $|S| \geq k$, of pairwise non-adjacent vertices?
- Vertex k-Coloring: Can the vertices of a graph be colored with $\leq k$ colors, so that any two adjacent vertices get different colors?

These three problems are NP-hard, but are they equally hard?

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n))$

- k-Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k)$

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

- k-Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

  The problem is FPT (fixed-parameter tractable)

- k-Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- *k*-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

  The problem is FPT (fixed-parameter tractable)

- *k*-Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

  The problem is XP (slice-wise polynomial)

- Vertex *k*-Coloring: NP-hard for fixed $k = 3$.

They behave quite differently...

- k-Vertex Cover: Solvable in time $\mathcal{O}(2^k \cdot (m+n)) = f(k) \cdot n^{\mathcal{O}(1)}$.

  The problem is FPT (fixed-parameter tractable)

- k-Independent Set: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

  The problem is XP (slice-wise polynomial)

- Vertex k-Coloring: NP-hard for fixed $k = 3$.

  The problem is para-NP-hard

Why k-CLIQUE may not be FPT?

k-INDEPENDENT SET: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

Why k-Clique may not be FPT?

k-Clique: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

Why k-Clique may not be FPT?

k-Clique: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

Why k-Clique may not be FPT?

Why k-Clique may not be FPT?

k-Clique: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

Why k-Clique may not be FPT?

So far, nobody has managed to find an FPT algorithm.

(also, nobody has found a poly-time algorithm for 3-SAT)

Why k-Clique may not be FPT?

k-Clique: Solvable in time $\mathcal{O}(k^2 \cdot n^k) = f(k) \cdot n^{g(k)}$.

Why k-Clique may not be FPT?

So far, nobody has managed to find an FPT algorithm.

(also, nobody has found a poly-time algorithm for 3-SAT)

Working hypothesis of parameterized complexity: k-Clique is not FPT

(in classical complexity: 3-SAT cannot be solved in poly-time)

How to transfer hardness among parameterized problems?

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

How to transfer hardness among parameterized problems?

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

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

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

How to transfer hardness among parameterized problems?

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

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

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

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

How to transfer hardness among parameterized problems?

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

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

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

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

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

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

How to transfer hardness among parameterized problems?

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

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

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

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

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

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

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

How to transfer hardness among parameterized problems?

Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems.

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

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

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

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

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

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

Kernelization

Idea polynomial-time preprocessing.

Kernelization

Idea polynomial-time preprocessing.

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

Kernelization

Idea polynomial-time preprocessing.

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

Instance (x, k) of A → polynomial time → Instance (x', k') of A

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

Kernelization

Idea polynomial-time preprocessing.

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

Instance (x, k) of A → polynomial time → Instance (x', k') of A

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

The function g is called the size of the kernel.

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

Kernelization

Idea: polynomial-time preprocessing.

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

Instance (x, k) of A → polynomial time → Instance (x', k') of A

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

The function g is called the size of the kernel.

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

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

Do all FPT problems admit polynomial kernels?

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

Do all FPT problems admit polynomial kernels?

Do all FPT problems admit polynomial kernels?

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

Do all FPT problems admit polynomial kernels? NO!

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

Deciding whether a graph has a PATH *with* $\geq k$ *vertices* is FPT *but does not admit a polynomial kernel, unless* NP $\subseteq$ coNP/poly.

Typical approach to deal with a parameterized problem

Parameterized problem L

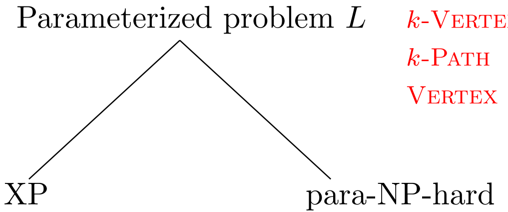
k-Clique

k-Vertex Cover

k-Path

Vertex k-Coloring

Typical approach to deal with a parameterized problem



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

Typical approach to deal with a parameterized problem

Parameterized problem L

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

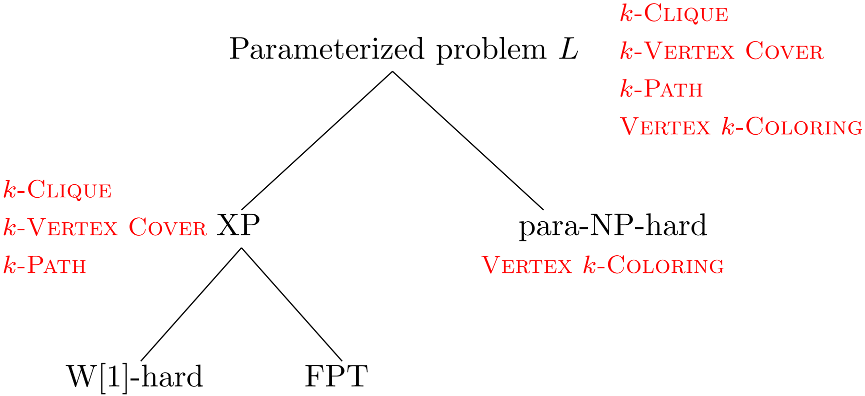
XP

k-Clique
k-Vertex Cover
k-Path

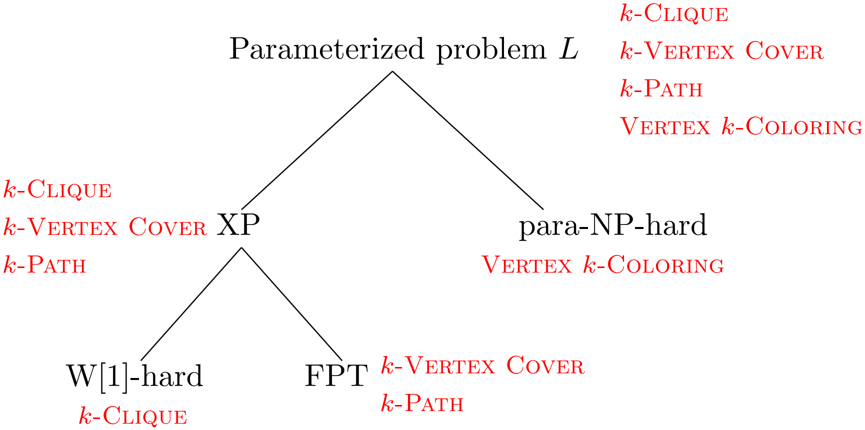
para-NP-hard

Vertex k-Coloring

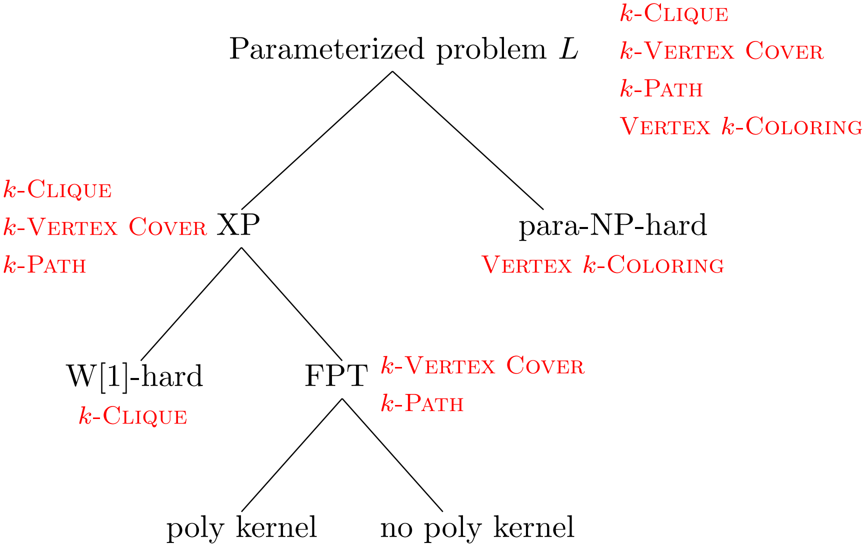
Typical approach to deal with a parameterized problem



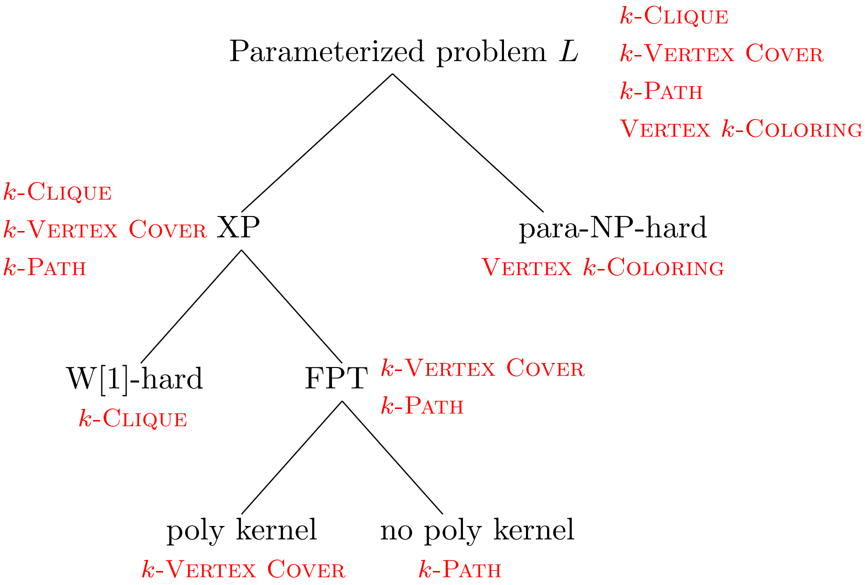
Typical approach to deal with a parameterized problem



Typical approach to deal with a parameterized problem



Typical approach to deal with a parameterized problem



Next subsection is...

1. Introduction
   - Career path
   - Scientific context: parameterized complexity
   - A relevant parameter: treewidth

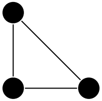
2. Some of my contributions (related to treewidth)
   - The number of graphs of bounded treewidth
   - Linear kernels on sparse graphs
   - Fast FPT algorithms parameterized by treewidth

3. Conclusions

Treewidth via k-trees

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

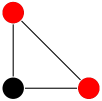
Example of a 2-tree:



[Figure by Julien Baste]

Treewidth via k-trees

Example of a 2-tree:

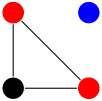


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

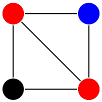


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

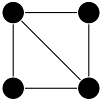


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

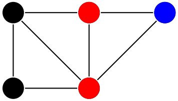


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

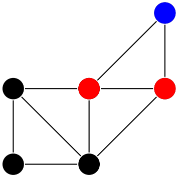


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

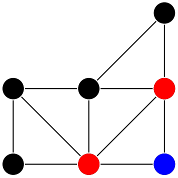


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

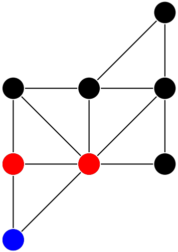


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

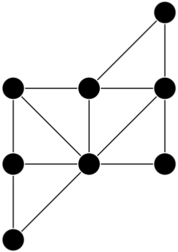


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:

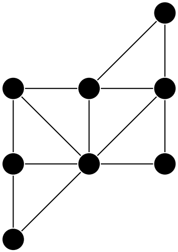


[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

Treewidth via k-trees

Example of a 2-tree:



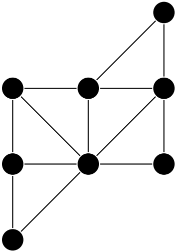
[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

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

Treewidth via k-trees

Example of a 2-tree:



[Figure by Julien Baste]

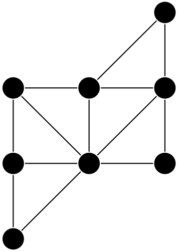
A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

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

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

Treewidth via k-trees

Example of a 2-tree:



[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

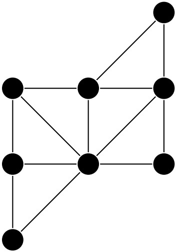
A partial k-tree is a subgraph of a k-tree.

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

Invariant that measures the topological resemblance of a graph to a tree.

Treewidth via k-trees

Example of a 2-tree:



[Figure by Julien Baste]

A k-tree is a graph that can be built starting from a $(k+1)$-clique and then iteratively adding a vertex connected to a k-clique.

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

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

Invariant that measures the topological resemblance of a graph to a tree.

Construction suggests the notion of tree decomposition: small separators.

Why treewidth?

Treewidth is important for (at least) 3 different reasons:

Why treewidth?

Treewidth is important for (at least) 3 different reasons:

1. Treewidth is a fundamental combinatorial tool in graph theory: key role in the Graph Minors project of Robertson and Seymour.

Why treewidth?

Treewidth is important for (at least) 3 different reasons:

1. Treewidth is a fundamental combinatorial tool in graph theory: key role in the Graph Minors project of Robertson and Seymour.

2. Treewidth behaves very well algorithmically, and algorithms parameterized by treewidth appear very often in FPT algorithms.

Why treewidth?

Treewidth is important for (at least) 3 different reasons:

1. Treewidth is a fundamental combinatorial tool in graph theory: key role in the Graph Minors project of Robertson and Seymour.

2. Treewidth behaves very well algorithmically, and algorithms parameterized by treewidth appear very often in FPT algorithms.

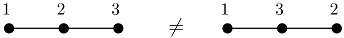
3. In many practical scenarios, it turns out that the treewidth of the associated graph is small (programming languages, road networks, ...).

Next section is...

1. Introduction
   - Career path
   - Scientific context: parameterized complexity
   - A relevant parameter: treewidth
2. Some of my contributions (related to treewidth)
   - The number of graphs of bounded treewidth
   - Linear kernels on sparse graphs
   - Fast FPT algorithms parameterized by treewidth
3. Conclusions

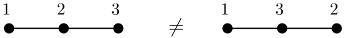
Next subsection is...

What is known about the number of (partial) k-trees?



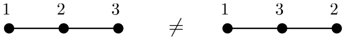
What is known about the number of (partial) k-trees?

Labeled k-trees



What is known about the number of (partial) k-trees?

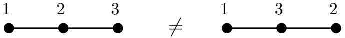
Labeled k-trees



- The number of n-vertex labeled trees is n^{n-2}. [Cayley. 1889]

What is known about the number of (partial) k-trees?

Labeled k-trees

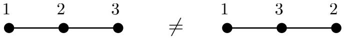


- The number of n-vertex labeled trees is n^{n-2}. [Cayley. 1889]
- The number of n-vertex labeled k-trees is $\binom{n}{k}(kn - k^2 + 1)^{n-k-2}$.

[Beineke, Pippert. 1969]

What is known about the number of (partial) k-trees?

Labeled k-trees

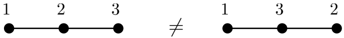


- The number of n-vertex labeled trees is n^{n-2}. [Cayley. 1889]
- The number of n-vertex labeled k-trees is $\binom{n}{k}(kn - k^2 + 1)^{n-k-2}$. [Beineke, Pippert. 1969]

Labeled partial k-trees

What is known about the number of (partial) k-trees?

Labeled k-trees



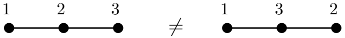
- The number of n-vertex labeled trees is n^{n-2}. [Cayley. 1889]
- The number of n-vertex labeled k-trees is $\binom{n}{k}(kn - k^2 + 1)^{n-k-2}$. [Beineke, Pippert. 1969]

Labeled partial k-trees

- $k = 1$: The number of n-vertex labeled forests is $\sim c \cdot n^{n-2}$ for some explicit constant $c > 1$. [Takács. 1990]

What is known about the number of (partial) k-trees?

Labeled k-trees



- The number of n-vertex labeled trees is n^{n-2}. [Cayley. 1889]
- The number of n-vertex labeled k-trees is $\binom{n}{k}(kn - k^2 + 1)^{n-k-2}$.

  [Beineke, Pippert. 1969]

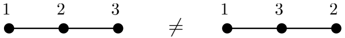
Labeled partial k-trees

- $k = 1$: The number of n-vertex labeled forests is $\sim c \cdot n^{n-2}$ for some explicit constant $c > 1$. [Takács. 1990]
- $k = 2$: The number of n-vertex labeled series-parallel graphs is $\sim g \cdot n^{-\frac{5}{2}} \gamma^n n!$ for some constants $g, \gamma > 0$.

  [Bodirsky, Giménez, Kang, Noy. 2005]

What is known about the number of (partial) k-trees?

Labeled k-trees



- The number of n-vertex labeled trees is n^{n-2}. [Cayley. 1889]
- The number of n-vertex labeled k-trees is $\binom{n}{k}(kn - k^2 + 1)^{n-k-2}$.

  [Beineke, Pippert. 1969]

Labeled partial k-trees

- $k = 1$: The number of n-vertex labeled forests is $\sim c \cdot n^{n-2}$ for some explicit constant $c > 1$. [Takács. 1990]
- $k = 2$: The number of n-vertex labeled series-parallel graphs is $\sim g \cdot n^{-\frac{5}{2}} \gamma^n n!$ for some constants $g, \gamma > 0$.

  [Bodirsky, Giménez, Kang, Noy. 2005]
- Nothing was known for general k.

$T_{n,k}$ and an easy upper bound

Let $T_{n,k}$ be the number of n-vertex labeled partial k-trees.

Objective Obtaining accurate bounds for $T_{n,k}$.

$T_{n,k}$ and an easy upper bound

Let $T_{n,k}$ be the number of n-vertex labeled partial k-trees.

Objective Obtaining accurate bounds for $T_{n,k}$.

As an n-vertex k-tree has $kn - \frac{k(k+1)}{2}$ edges, we get the upper bound:

$$T_{n,k} \leq \binom{n}{k} \cdot (kn - k^2 + 1)^{n-k-2} \cdot 2^{kn - \frac{k(k+1)}{2}}$$

$T_{n,k}$ and an easy upper bound

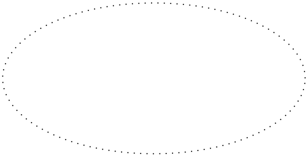
Let $T_{n,k}$ be the number of n-vertex labeled partial k-trees.

Objective Obtaining accurate bounds for $T_{n,k}$.

As an n-vertex k-tree has $kn - \frac{k(k+1)}{2}$ edges, we get the upper bound:

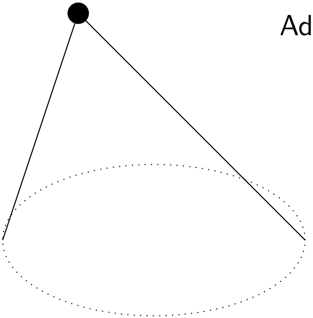
$$
\begin{aligned}
T_{n,k} &\leq \binom{n}{k} \cdot (kn - k^2 + 1)^{n-k-2} \cdot 2^{kn - \frac{k(k+1)}{2}} \\
&\leq (k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot k^{-k}
\end{aligned}
$$

An easy lower bound



Take a forest on $n - (k - 1)$ vertices:
$(n - k + 1)^{(n-k-1)}$ possibilities

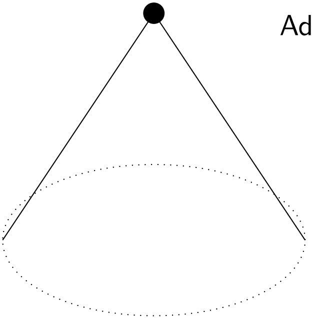
An easy lower bound



Add a vertex arbitrarily connected to the forest:
$2^{n-(k-1)}$ possibilities

Take a forest on $n-(k-1)$ vertices:
$(n-k+1)^{(n-k-1)}$ possibilities

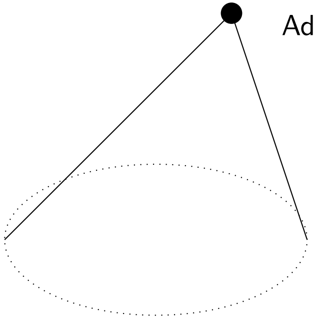
An easy lower bound



Add a vertex arbitrarily connected to the forest:
$2^{n-(k-1)}$ possibilities

Take a forest on $n-(k-1)$ vertices:
$(n-k+1)^{(n-k-1)}$ possibilities

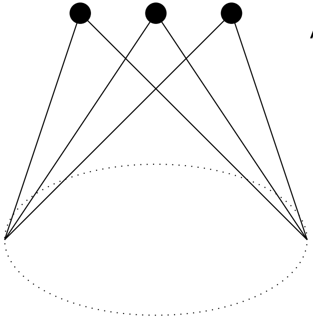
An easy lower bound



Add a vertex arbitrarily connected to the forest:
$2^{n-(k-1)}$ possibilities

Take a forest on $n-(k-1)$ vertices:
$(n-k+1)^{(n-k-1)}$ possibilities

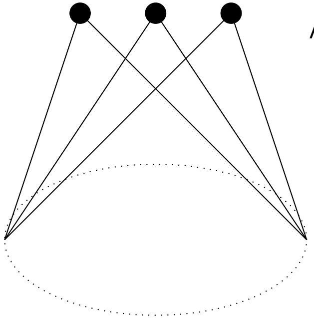
An easy lower bound



Add $k-1$ vertices connected to the forest:
$\geq 2^{(k-1)(n-(k-1))}$ possibilities

Take a forest on $n-(k-1)$ vertices:
$(n-k+1)^{(n-k-1)}$ possibilities

An easy lower bound

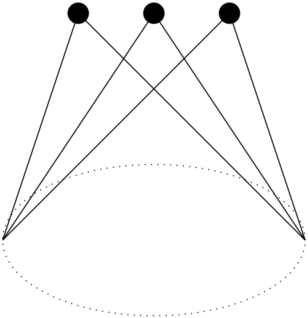


Add $k-1$ vertices connected to the forest: $\geq 2^{(k-1)(n-(k-1))}$ possibilities

Take a forest on $n-(k-1)$ vertices: $(n-k+1)^{(n-k-1)}$ possibilities

$$T_{n,k} \;\geq\; (n-k+1)^{(n-k-1)} \cdot 2^{(k-1)(n-k+1)}$$

An easy lower bound



Add $k-1$ vertices connected to the forest:
$\geq 2^{(k-1)(n-(k-1))}$ possibilities

Take a forest on $n-(k-1)$ vertices:
$(n-k+1)^{(n-k-1)}$ possibilities

$$T_{n,k} \geq (n-k+1)^{(n-k-1)} \cdot 2^{(k-1)(n-k+1)} \geq \left(\frac{1}{4} \cdot 2^k \cdot n\right)^n \cdot 2^{-k^2}$$

An improved lower bound

Summarizing, so far we have:

$$T_{n,k} \leq (k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot k^{-k}$$

$$T_{n,k} \geq \left(\frac{1}{4} \cdot 2^k \cdot n\right)^n \cdot 2^{-k^2}$$

An improved lower bound

Summarizing, so far we have:

$$T_{n,k} \leq (k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot k^{-k}$$

$$T_{n,k} \geq \left(\frac{1}{4} \cdot 2^k \cdot n\right)^n \cdot 2^{-k^2}$$

Gap in the dominant term: $\boxed{(4 \cdot k)^n}$

An improved lower bound

Summarizing, so far we have:

$$T_{n,k} \leq (k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot k^{-k}$$

$$T_{n,k} \geq \left(\frac{1}{4} \cdot 2^k \cdot n\right)^n \cdot 2^{-k^2}$$

Gap in the dominant term: $\boxed{(4 \cdot k)^n}$

Theorem (Baste, Noy, S., 2017)

For any two integers n, k with $1 < k \leq n$, the number $T_{n,k}$ of n-vertex labeled graphs with treewidth at most k satisfies

$$T_{n,k} \geq \left(\frac{1}{128e} \cdot \frac{k}{\log k} \cdot 2^k \cdot n\right)^n \cdot 2^{-\frac{k(k+3)}{2}} \cdot k^{-2k-2}.$$

An improved lower bound

Summarizing, so far we have:

$$T_{n,k} \leq (k \cdot 2^k \cdot n)^n \cdot 2^{-\frac{k(k+1)}{2}} \cdot k^{-k}$$

$$T_{n,k} \geq \left(\frac{1}{4} \cdot 2^k \cdot n\right)^n \cdot 2^{-k^2}$$

Gap in the dominant term: $\boxed{(4 \cdot k)^n}$

Theorem (Baste, Noy, S., 2017)

For any two integers n, k with $1 < k \leq n$, the number $T_{n,k}$ of n-vertex labeled graphs with treewidth at most k satisfies

$$T_{n,k} \geq \left(\frac{1}{128e} \cdot \frac{k}{\log k} \cdot 2^k \cdot n\right)^n \cdot 2^{-\frac{k(k+3)}{2}} \cdot k^{-2k-2}.$$

Gap in the dominant term: $\boxed{(128e \cdot \log k)^n}$

Next subsection is...

Meta-kernelization

As in the case of FPT algorithms, there exist meta-kernelization results.

Meta-kernelization

As in the case of FPT algorithms, there exist meta-kernelization results.

Typical statement:

Every parameterized problem that satisfies property Π is admits a linear/polynomial kernel on the class of graphs $\mathcal{G}$.

Meta-kernelization

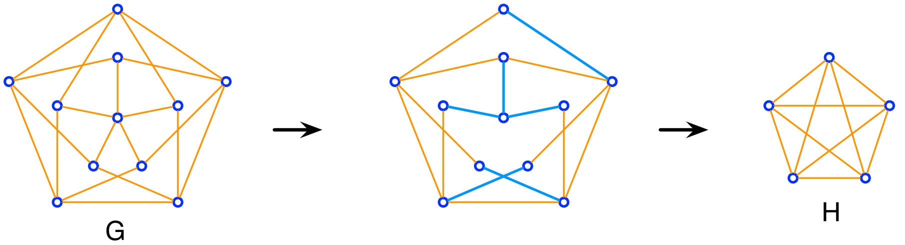
As in the case of FPT algorithms, there exist meta-kernelization results.

Typical statement:

Every parameterized problem that satisfies property Π is admits a linear/polynomial kernel on the class of graphs $\mathcal{G}$.

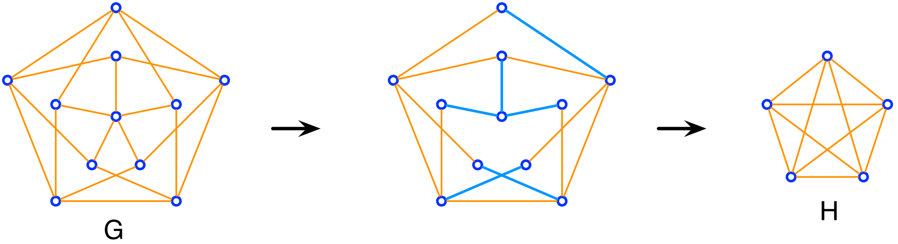
This has been also a very active area in parameterized complexity, specially on sparse graphs: planar graphs, graphs on surfaces, minor-free graphs, ...

Minors and topological minors



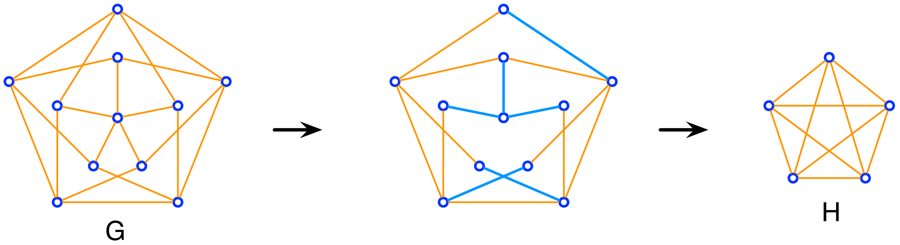
- H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges.

Minors and topological minors



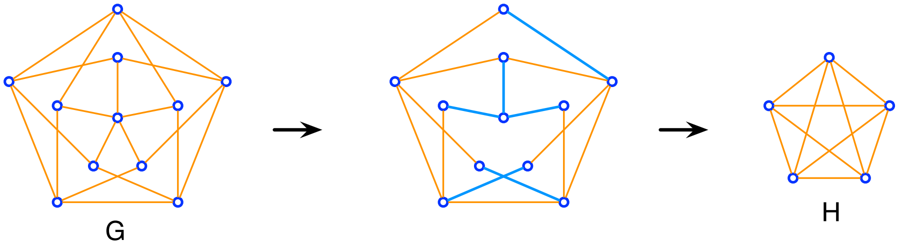
- H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges.
- H is a topological minor of G if H can be obtained from a subgraph of G by contracting edges with at least one endpoint of deg ≤ 2.

Minors and topological minors



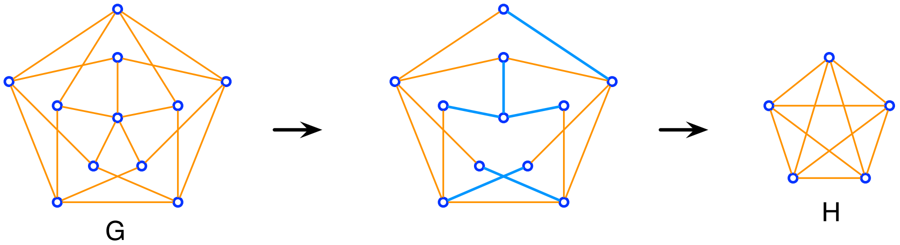
- H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges.
- H is a topological minor of G if H can be obtained from a subgraph of G by contracting edges with at least one endpoint of deg ≤ 2.
- Therefore: H topological minor of $G \Rightarrow H$ minor of G

Minors and topological minors



- H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges.
- H is a topological minor of G if H can be obtained from a subgraph of G by contracting edges with at least one endpoint of deg ≤ 2.
- Therefore: H topological minor of G $\not\Leftarrow$ H minor of G

Minors and topological minors



- H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges.
- H is a topological minor of G if H can be obtained from a subgraph of G by contracting edges with at least one endpoint of deg ≤ 2.
- Therefore: H topological minor of G $\not\Leftarrow$ H minor of G
- Fixed H: H-minor-free graphs $\subseteq$ H-topological-minor-free graphs

Linear kernels on sparse graphs – an overview

- DOMINATING SET on planar graphs. [Alber, Fellows, Niedermeier. 2002]

Linear kernels on sparse graphs – an overview

- DOMINATING SET on planar graphs. [Alber, Fellows, Niedermeier. 2002]
- Framework for several problems on planar graphs. [Guo, Niedermeier. 2007]

Linear kernels on sparse graphs – an overview

- DOMINATING SET on planar graphs. [Alber, Fellows, Niedermeier. 2002]
- Framework for several problems on planar graphs. [Guo, Niedermeier. 2007]
- Meta-kernelization for graphs of bounded genus. [Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]

Linear kernels on sparse graphs – an overview

- Dominating Set on planar graphs. [Alber, Fellows, Niedermeier. 2002]
- Framework for several problems on planar graphs. [Guo, Niedermeier. 2007]
- Meta-kernelization for graphs of bounded genus. [Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]
- Meta-kernelization for minor-free graphs. [Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

Linear kernels on sparse graphs – an overview

- DOMINATING SET on planar graphs. [Alber, Fellows, Niedermeier. 2002]
- Framework for several problems on planar graphs. [Guo, Niedermeier. 2007]
- Meta-kernelization for graphs of bounded genus. [Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]
- Meta-kernelization for minor-free graphs. [Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

★ Meta-kernelization for topological-minor-free graphs.

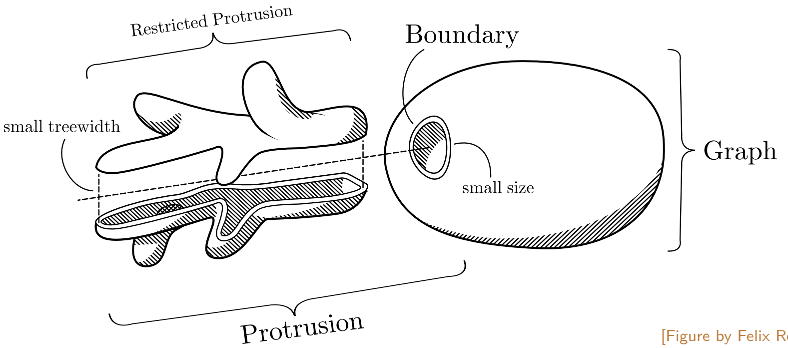
[Kim, Langer, Paul, Reidl, Rossmanith, **S**., Sikdar. 2013]

Protrusions

[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]

- Given a graph G, a set $W \subseteq V(G)$ is a t-protrusion of G if

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



[Figure by Felix Reidl]

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

Meta-kernelization on graphs without topological minors

Theorem (Kim, Langer, Paul, Reidl, Rossmanith, S., Sikdar, 2013)

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

Meta-kernelization on graphs without topological minors

Theorem (Kim, Langer, Paul, Reidl, Rossmanith, S., Sikdar, 2013)

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

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

$$\exists X \subseteq V(G) \text{ s.t. } |X| \leq c \cdot k \text{ and } \mathrm{tw}(G - X) \leq t.$$

Meta-kernelization on graphs without topological minors

Theorem (Kim, Langer, Paul, Reidl, Rossmanith, S., Sikdar, 2013)

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

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

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

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

Meta-kernelization on graphs without topological minors

Theorem (Kim, Langer, Paul, Reidl, Rossmanith, S., Sikdar, 2013)

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

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

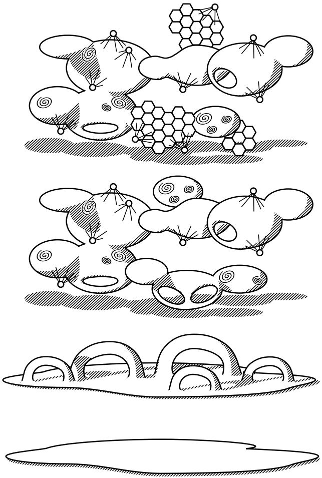
$$\exists X \subseteq V(G) \text{ s.t. } |X| \leq c \cdot k \text{ and } \mathrm{tw}(G - X) \leq t.$$

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

Some problems affected by our result:

TREEWIDTH-t VERTEX DELETION, CHORDAL VERTEX DELETION, INTERVAL VERTEX DELETION, EDGE DOMINATING SET, FEEDBACK VERTEX SET, CONNECTED VERTEX COVER, ...

Linear kernels on sparse graphs – the conditions



H-topological-minor-free — treewidth-bounding

$\cup$

H-minor-free — bidimensional, separation property

$\cup$

bounded genus — quasi-compact

$\cup$

planar — "distance-property"

[Figure by Felix Reidl]

Are our conditions very restrictive?

We require FII + treewidth-bounding

Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

Are our conditions very restrictive?

We require FII + treewidth-bounding

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

Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

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

  Conditions on H-minor-free graphs:
  bidimensional + separation property. [Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

Are our conditions very restrictive?

We require FII + treewidth-bounding

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

  Conditions on H-minor-free graphs:
  bidimensional + separation property. [Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

  But it holds that

  bidimensional + separation property $\Rightarrow$ treewidth-bounding

Are our conditions very restrictive?

We require FII + treewidth-bounding

- FII is necessary when using protrusion replacement rules.

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

  Conditions on H-minor-free graphs:
  bidimensional + separation property. [Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

  But it holds that

  bidimensional + separation property $\Rightarrow$ treewidth-bounding

- Our results imply the linear kernels of [Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

Idea of the proof(s): protrusion decompositions

Idea of the proof(s): protrusion decompositions

Given a problem P with parameter k, a linear protrusion decomposition of a graph G is a partition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:

Idea of the proof(s): protrusion decompositions

Given a problem P with parameter k, a linear protrusion decomposition of a graph G is a partition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:

- $|Y_0| = \mathcal{O}(k)$.

Idea of the proof(s): protrusion decompositions

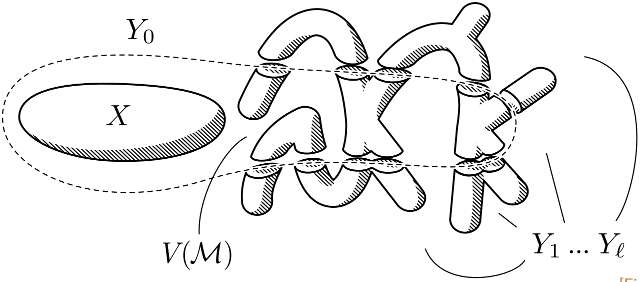
Given a problem P with parameter k, a linear protrusion decomposition of a graph G is a partition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:

- $|Y_0| = \mathcal{O}(k)$.
- $\ell = \mathcal{O}(k)$ and for every $1 \leqslant i \leqslant \ell$, Y_i is a t-protrusion.

Idea of the proof(s): protrusion decompositions

Given a problem P with parameter k, a linear protrusion decomposition of a graph G is a partition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:

- $|Y_0| = \mathcal{O}(k)$.
- $\ell = \mathcal{O}(k)$ and for every $1 \leqslant i \leqslant \ell$, Y_i is a t-protrusion.

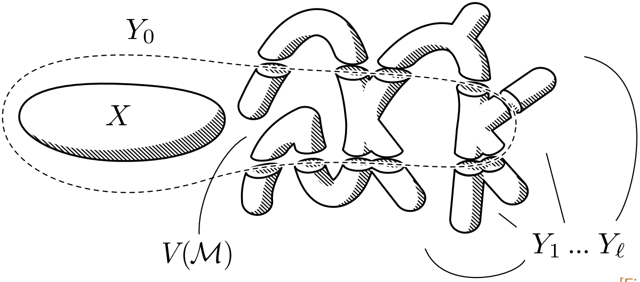


[Figure by Felix Reidl]

Idea of the proof(s): protrusion decompositions

Given a problem P with parameter k, a linear protrusion decomposition of a graph G is a partition $Y_0 \uplus Y_1 \uplus \cdots \uplus Y_\ell$ of $V(G)$ such that:

- $|Y_0| = \mathcal{O}(k)$.
- $\ell = \mathcal{O}(k)$ and for every $1 \leqslant i \leqslant \ell$, Y_i is a t-protrusion.



[Figure by Felix Reidl]

Idea: Find a linear protrusion decomposition in polynomial time, and replace each of the $\mathcal{O}(k)$ protrusions with a constant-sized gadget.

Constructibility issues

Idea Find a linear protrusion decomposition in polynomial time, and replace each of the $\mathcal{O}(k)$ protrusions with a constant-sized gadget.

Constructibility issues

Idea Find a linear protrusion decomposition in polynomial time, and replace each of the $\mathcal{O}(k)$ protrusions with a constant-sized gadget.

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

Constructibility issues

Idea Find a linear protrusion decomposition in polynomial time, and replace each of the $\mathcal{O}(k)$ protrusions with a constant-sized gadget.

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

Same problem in the previous work based on protrusion replacement:

[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]

[Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

Constructibility issues

Idea Find a linear protrusion decomposition in polynomial time, and replace each of the $\mathcal{O}(k)$ protrusions with a constant-sized gadget.

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

Same problem in the previous work based on protrusion replacement:

[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos. 2009]

[Fomin, Lokshtanov, Saurabh, Thilikos. 2010]

There are some techniques to actually construct the kernels (CMSO logic), but it is hard to extract explicit constants on the size of the kernels...

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

Instead of FII or CMSO logic

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

Instead of FII or CMSO logic → we use → dynamic programming

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

Instead of FII or CMSO logic → we use → dynamic programming

- We formalize the notion of encoding for the tables of dynamic programming (DP) on tree decompositions.

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

Instead of FII or CMSO logic → we use → dynamic programming

- We formalize the notion of encoding for the tables of dynamic programming (DP) on tree decompositions.
- Conditions to obtain protrusion replacer with explicit constants:

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

Instead of FII or CMSO logic → we use → dynamic programming

- We formalize the notion of encoding for the tables of dynamic programming (DP) on tree decompositions.
- Conditions to obtain protrusion replacer with explicit constants:

Confined encoder: number of distinct values is a function of the tw.

Explicit linear kernels via dynamic programming

We propose an approach to replace protrusions with explicit constants:

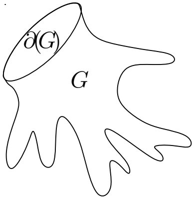
Instead of FII or CMSO logic — we use → dynamic programming

- We formalize the notion of encoding for the tables of dynamic programming (DP) on tree decompositions.
- Conditions to obtain protrusion replacer with explicit constants:

  Confined encoder: number of distinct values is a function of the tw.

  DP-friendly encoder: we can safely replace equivalent "protrusions".

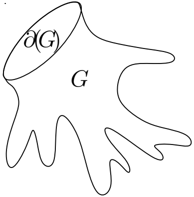
An explicit meta-kernelization result



[By Valentin Garnero]

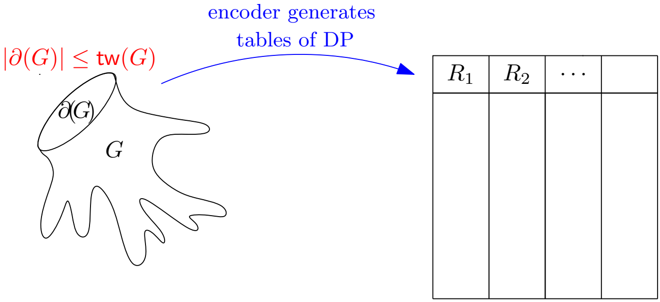
An explicit meta-kernelization result

$|\partial(G)| \leq \mathsf{tw}(G)$



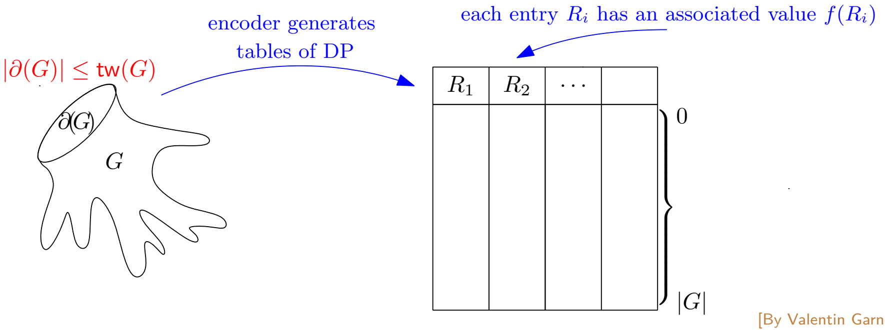
[By Valentin Garnero]

An explicit meta-kernelization result



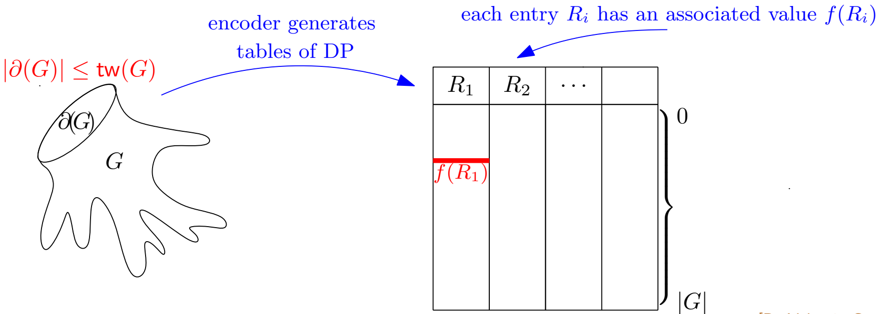
[By Valentin Garnero]

An explicit meta-kernelization result



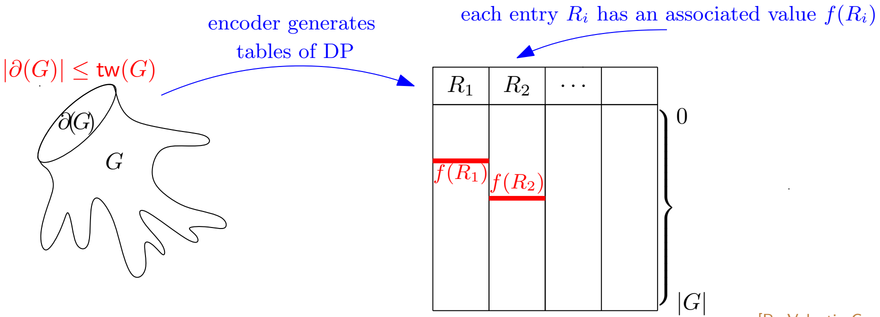
[By Valentin Garnero]

An explicit meta-kernelization result



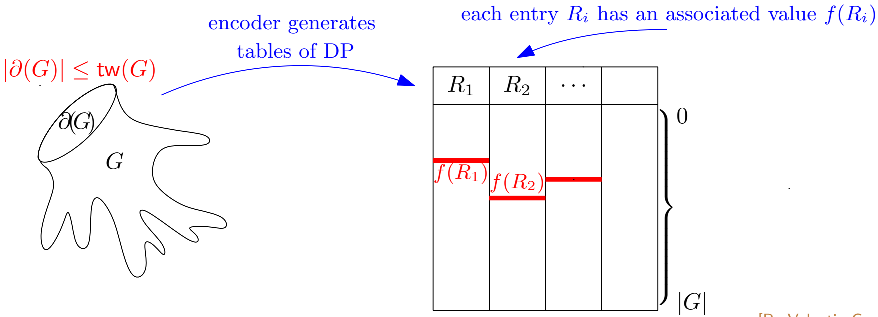
[By Valentin Garnero]

An explicit meta-kernelization result



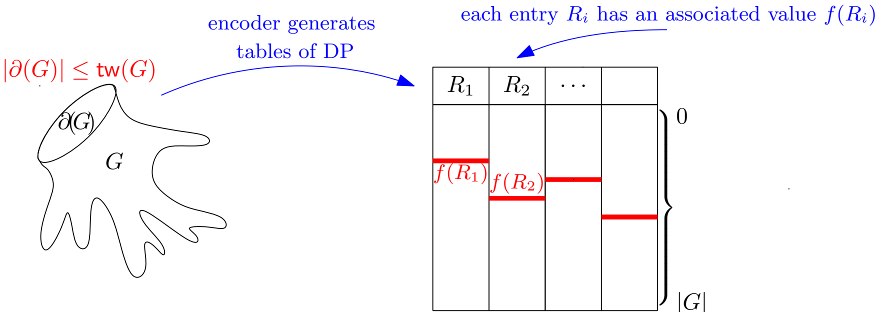
[By Valentin Garnero]

An explicit meta-kernelization result



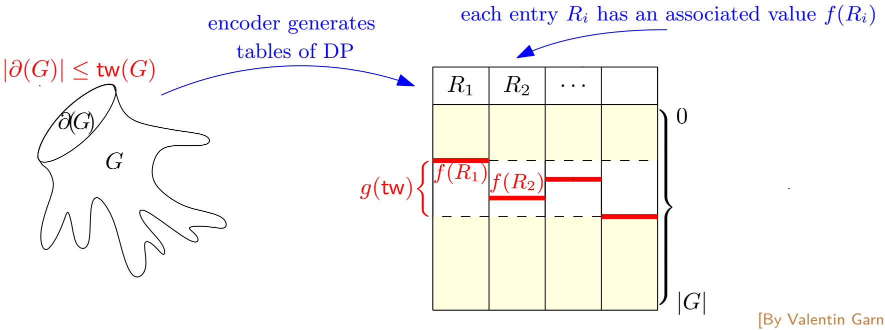
[By Valentin Garnero]

An explicit meta-kernelization result



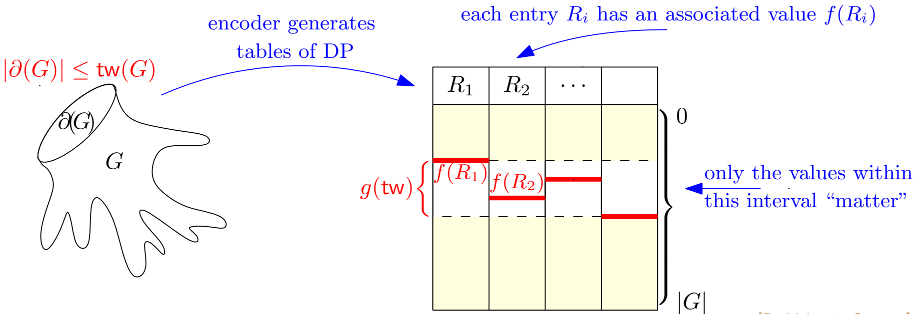
[By Valentin Garnero]

An explicit meta-kernelization result



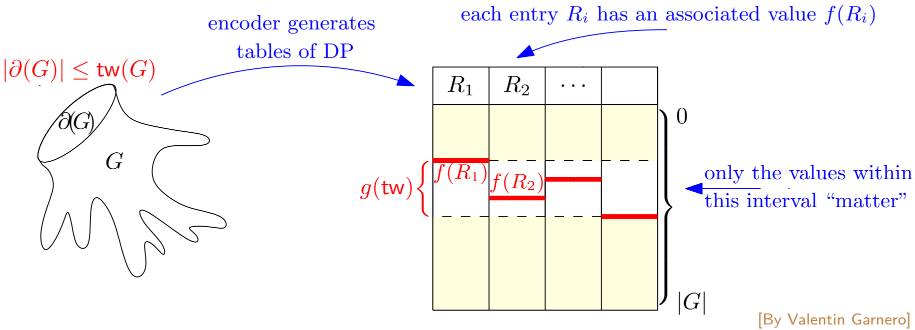
[By Valentin Garnero]

An explicit meta-kernelization result



[By Valentin Garnero]

An explicit meta-kernelization result

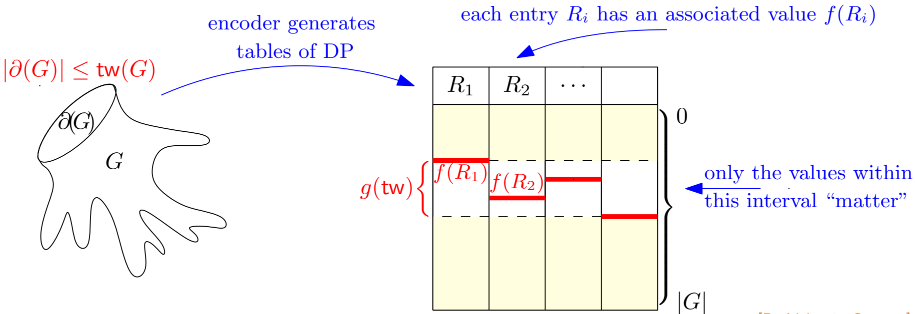


[By Valentin Garnero]

Theorem (Garnero, Paul, S., Thilikos, 2014)

Linear protrusion decomp. $+$ *confined DP-friendly encoder* $=$ *linear kernel.*

An explicit meta-kernelization result



[By Valentin Garnero]

Theorem (Garnero, Paul, S., Thilikos, 2014)

Linear protrusion decomp. + *confined DP-friendly encoder* = *linear kernel.*

Some problems affected by our result:

r-Dominating Set and r-Scattered Set on apex-minor-free graphs,
Planar-$\mathcal{F}$-Deletion on (topological)-minor-free graphs,
several generalizations of $\mathcal{F}$-Packing, ...

Next subsection is...

1. Introduction
   - Career path
   - Scientific context: parameterized complexity
   - A relevant parameter: treewidth
2. Some of my contributions (related to treewidth)
   - The number of graphs of bounded treewidth
   - Linear kernels on sparse graphs
   - Fast FPT algorithms parameterized by treewidth
3. Conclusions

Treewidth behaves very well algorithmically

Treewidth behaves very well algorithmically

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

Treewidth behaves very well algorithmically

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

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

Treewidth behaves very well algorithmically

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

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

Theorem (Courcelle, 1990)

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

Examples: Vertex Cover, Dominating Set, Hamiltonian Cycle, Clique, Independent Set, k-Coloring for fixed k, ...

Is it enough to prove that a problem is FPT?

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

$$f(\mathrm{tw}) \cdot n^{\mathcal{O}(1)}$$

Is it enough to prove that a problem is FPT?

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

... but the running time can (and must) be huge!

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

Is it enough to prove that a problem is FPT?

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

... but the running time can (and must) be huge!

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

Major goal: find the smallest possible function $f(\text{tw})$.

This is a very active area in parameterized complexity.

Lower bounds on the running times of FPT algorithms

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

Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

ETH: The 3-SAT problem on n variables cannot be solved in time $2^{o(n)}$

SETH: The SAT problem on n variables cannot be solved in time $(2-\varepsilon)^n$

[Impagliazzo, Paturi. 1999]

Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

ETH: The 3-SAT problem on n variables cannot be solved in time $2^{o(n)}$

SETH: The SAT problem on n variables cannot be solved in time $(2 - \varepsilon)^n$

[Impagliazzo, Paturi. 1999]

$$\text{SETH} \Rightarrow \text{ETH}$$

Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

ETH: The 3-SAT problem on n variables cannot be solved in time $2^{o(n)}$

SETH: The SAT problem on n variables cannot be solved in time $(2 - \varepsilon)^n$

[Impagliazzo, Paturi. 1999]

$$\text{SETH} \Rightarrow \text{ETH} \Rightarrow \text{FPT} \neq \text{W[1]}$$

Lower bounds on the running times of FPT algorithms

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

  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?

  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

ETH: The 3-SAT problem on n variables cannot be solved in time $2^{o(n)}$

SETH: The SAT problem on n variables cannot be solved in time $(2-\varepsilon)^n$

[Impagliazzo, Paturi. 1999]

$$\mathrm{SETH} \Rightarrow \mathrm{ETH} \Rightarrow \mathsf{FPT} \neq \mathsf{W}[1] \Rightarrow \mathsf{P} \neq \mathsf{NP}$$

Lower bounds on the running times of FPT algorithms

- Suppose that we have an FPT algorithm in time $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$?
  Is it possible to obtain an FPT algorithm in time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$?

Very helpful tool: (Strong) Exponential Time Hypothesis – (S)ETH

ETH: The 3-SAT problem on n variables cannot be solved in time $2^{o(n)}$

SETH: The SAT problem on n variables cannot be solved in time $(2-\varepsilon)^n$

[Impagliazzo, Paturi. 1999]

$$\text{SETH} \Rightarrow \text{ETH} \Rightarrow \text{FPT} \neq \text{W[1]} \Rightarrow \text{P} \neq \text{NP}$$

Typical statements:

ETH $\Rightarrow$ k-Vertex Cover cannot be solved in time $2^{o(k)} \cdot n^{O(1)}$.

ETH $\Rightarrow$ Planar k-Vertex Cover cannot in time $2^{o(\sqrt{k})} \cdot n^{O(1)}$.

Bounds for problems parameterized by treewidth

Typically, FPT algorithms parameterized by treewidth are based on dynamic programming (DP) over a tree decomposition.

Bounds for problems parameterized by treewidth

Typically, FPT algorithms parameterized by treewidth are based on dynamic programming (DP) over a tree decomposition.

For many problems, like Vertex Cover or Dominating Set, the "natural" DP algorithms lead to (optimal) single-exponential algorithms:

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

Bounds for problems parameterized by treewidth

Typically, FPT algorithms parameterized by treewidth are based on dynamic programming (DP) over a tree decomposition.

For many problems, like Vertex Cover or Dominating Set, the "natural" DP algorithms lead to (optimal) single-exponential algorithms:

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

But for the so-called connectivity problems, like Longest Path or Steiner Tree, the "natural" DP algorithms provide only time

$$2^{\mathcal{O}(\mathrm{tw} \cdot \log \mathrm{tw})} \cdot n^{\mathcal{O}(1)}.$$

Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$:

Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$:

- Planar graphs: [Dorn, Penninkx, Bodlaender, Fomin. 2005]

Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\text{tw})} \cdot n^{\mathcal{O}(1)}$:

- Planar graphs: [Dorn, Penninkx, Bodlaender, Fomin. 2005]
- Graphs on surfaces: [Dorn, Fomin, Thilikos. 2006] [Rué, **S.**, Thilikos. 2010]

Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$:

- Planar graphs: [Dorn, Penninkx, Bodlaender, Fomin. 2005]
- Graphs on surfaces: [Dorn, Fomin, Thilikos. 2006]
  [Rué, **S.**, Thilikos. 2010]
- Minor-free graphs: [Dorn, Fomin, Thilikos. 2008]
  [Rué, **S.**, Thilikos. 2012]

Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$:

- Planar graphs: [Dorn, Penninkx, Bodlaender, Fomin. 2005]
- Graphs on surfaces: [Dorn, Fomin, Thilikos. 2006]
  [Rué, **S.**, Thilikos. 2010]
- Minor-free graphs: [Dorn, Fomin, Thilikos. 2008]
  [Rué, **S.**, Thilikos. 2012]

Main idea special type of decomposition with nice topological properties:

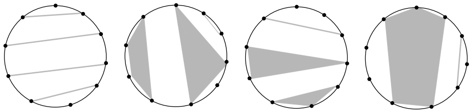
Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$:

- Planar graphs: [Dorn, Penninkx, Bodlaender, Fomin. 2005]
- Graphs on surfaces: [Dorn, Fomin, Thilikos. 2006]
  [Rué, **S.**, Thilikos. 2010]
- Minor-free graphs: [Dorn, Fomin, Thilikos. 2008]
  [Rué, **S.**, Thilikos. 2012]

Main idea special type of decomposition with nice topological properties:

partial solutions $\Longleftrightarrow$ non-crossing partitions



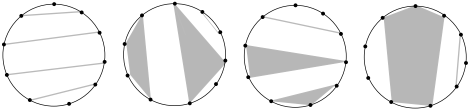
Single-exponential algorithms on sparse graphs

On topologically structured graphs (planar, surfaces, minor-free), it is possible to solve connectivity problems in time $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$:

- Planar graphs: [Dorn, Penninkx, Bodlaender, Fomin. 2005]
- Graphs on surfaces: [Dorn, Fomin, Thilikos. 2006]
  [Rué, **S.**, Thilikos. 2010]
- Minor-free graphs: [Dorn, Fomin, Thilikos. 2008]
  [Rué, **S.**, Thilikos. 2012]

Main idea special type of decomposition with nice topological properties:

partial solutions $\Longleftrightarrow$ non-crossing partitions



$$\mathrm{CN}(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi}k^{3/2}} \leq 4^k.$$

The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{\mathcal{O}(\mathrm{tw} \cdot \log \mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ were optimal for connectivity problems.

The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{\mathcal{O}(\mathrm{tw} \cdot \log \mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ were optimal for connectivity problems.

This was false!!

Cut&Count technique: [Cygan, Nederlof, Pilipczuk[2], van Rooij, Wojtaszczyk. 2011]
Randomized single-exponential algorithms for connectivity problems.

The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{\mathcal{O}(\mathrm{tw} \cdot \log \mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ were optimal for connectivity problems.

This was false!!

Cut&Count technique: [Cygan, Nederlof, Pilipczuk^2, van Rooij, Wojtaszczyk. 2011]

Randomized single-exponential algorithms for connectivity problems.

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

The revolution of single-exponential algorithms

It was believed that, except on sparse graphs (planar, surfaces), algorithms in time $2^{\mathcal{O}(\text{tw} \cdot \log \text{tw})} \cdot n^{\mathcal{O}(1)}$ were optimal for connectivity problems.

This was false!!

Cut&Count technique: [Cygan, Nederlof, Pilipczuk^2, van Rooij, Wojtaszczyk. 2011]
Randomized single-exponential algorithms for connectivity problems.

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

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

End of the story?

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

End of the story?

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

No!

Cycle Packing: find the maximum number of vertex-disjoint cycles.

An algorithm in time $2^{\mathcal{O}(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ is optimal under the ETH.

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. 2011]

End of the story?

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

No!

CYCLE PACKING: find the maximum number of vertex-disjoint cycles.

An algorithm in time $2^{\mathcal{O}(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ is optimal under the ETH.

[Cygan, Nederlof, Pilipczuk, Pilipczuk, van Rooij, Wojtaszczyk. 2011]

There are other examples of such problems...

The $\mathcal{F}$-DELETION problem

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

The $\mathcal{F}$-Deletion problem

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

$\mathcal{F}$-Deletion
Input: A graph G and an integer k.
Parameter: The treewidth tw of G.
Question: Does G contain a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

The $\mathcal{F}$-DELETION problem

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

$\mathcal{F}$-DELETION	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\|S\| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: VERTEX COVER.

The $\mathcal{F}$-Deletion problem

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

$\mathcal{F}$-Deletion	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\lvert S \rvert \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

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

The $\mathcal{F}$-Deletion problem

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

$\mathcal{F}$-Deletion	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\lvert S \rvert \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: Vertex Cover.
  Easily solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$.
- $\mathcal{F} = \{C_3\}$: Feedback Vertex Set.

The $\mathcal{F}$-Deletion problem

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

$\mathcal{F}$-Deletion	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\|S\| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: Vertex Cover.
  Easily solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$.
- $\mathcal{F} = \{C_3\}$: Feedback Vertex Set.
  "Hardly" solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$. [Cut&Count. 2011]

The $\mathcal{F}$-Deletion problem

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

$\mathcal{F}$-Deletion	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\lvert S\rvert \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: Vertex Cover.
  Easily solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$.
- $\mathcal{F} = \{C_3\}$: Feedback Vertex Set.
  "Hardly" solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$. [Cut&Count. 2011]
- $\mathcal{F} = \{K_5, K_{3,3}\}$: Vertex Planarization.

The $\mathcal{F}$-Deletion problem

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

$\mathcal{F}$-Deletion	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\lvert S\rvert \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: Vertex Cover.
  Easily solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$.
- $\mathcal{F} = \{C_3\}$: Feedback Vertex Set.
  "Hardly" solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$. [Cut&Count. 2011]
- $\mathcal{F} = \{K_5, K_{3,3}\}$: Vertex Planarization.
  Solvable in time $2^{\Theta(\text{tw} \cdot \log \text{tw})} \cdot n^{\mathcal{O}(1)}$. [Jansen, Lokshtanov, Saurabh. 2014 + Pilipczuk. 2015]

The $\mathcal{F}$-Deletion problem

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

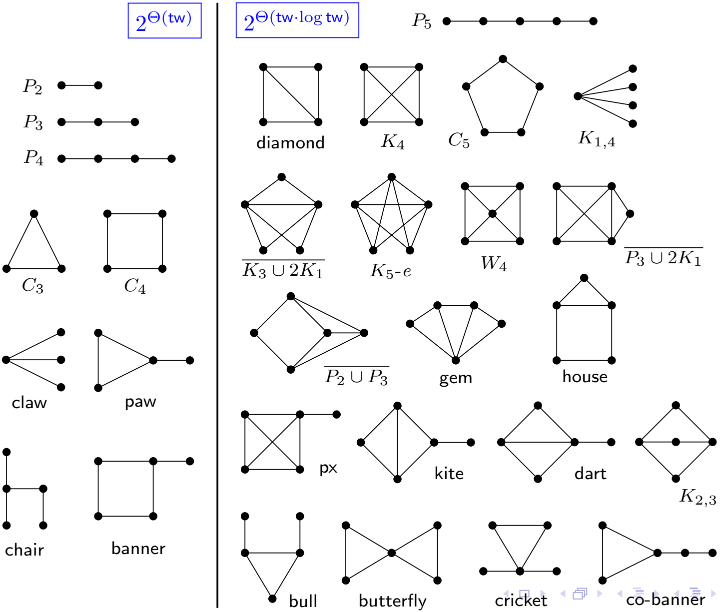
$\mathcal{F}$-Deletion	
Input:	A graph G and an integer k.
Parameter:	The treewidth tw of G.
Question:	Does G contain a set $S \subseteq V(G)$ with $\|S\| \leq k$ such that $G - S$ does not contain any of the graphs in $\mathcal{F}$ as a minor?

- $\mathcal{F} = \{K_2\}$: Vertex Cover.
  Easily solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$.
- $\mathcal{F} = \{C_3\}$: Feedback Vertex Set.
  "Hardly" solvable in time $2^{\Theta(\text{tw})} \cdot n^{\mathcal{O}(1)}$. [Cut&Count. 2011]
- $\mathcal{F} = \{K_5, K_{3,3}\}$: Vertex Planarization.
  Solvable in time $2^{\Theta(\text{tw} \cdot \log \text{tw})} \cdot n^{\mathcal{O}(1)}$. [Jansen, Lokshtanov, Saurabh. 2014 + Pilipczuk. 2015]

With Julien Baste and Dimitrios M. Thilikos we proved the following...

[arXiv:1704.07284. 2018]

Complexity of $\{H\}$-DELETION for small planar graphs H



Next section is...

1. Introduction
   - Career path
   - Scientific context: parameterized complexity
   - A relevant parameter: treewidth

2. Some of my contributions (related to treewidth)
   - The number of graphs of bounded treewidth
   - Linear kernels on sparse graphs
   - Fast FPT algorithms parameterized by treewidth

3. Conclusions

What's next?

There are many problems that I would like to solve...

In particular, several questions concerning $\mathcal{F}$-DELETION:

What's next?

There are many problems that I would like to solve...

In particular, several questions concerning $\mathcal{F}$-DELETION:

- Ultimate goal: classify the (asymptotically) tight complexity of $\mathcal{F}$-DELETION for every family $\mathcal{F}$

What's next?

There are many problems that I would like to solve...

In particular, several questions concerning $\mathcal{F}$-Deletion:

- Ultimate goal: classify the (asymptotically) tight complexity of $\mathcal{F}$-Deletion for every family $\mathcal{F}$... we are still very far from it.

What's next?

There are many problems that I would like to solve...

In particular, several questions concerning $\mathcal{F}$-DELETION:

- Ultimate goal: classify the (asymptotically) tight complexity of $\mathcal{F}$-DELETION for every family $\mathcal{F}$... we are still very far from it.
- We do not even know if there exists some $\mathcal{F}$ such that $\mathcal{F}$-DELETION cannot be solved in time $2^{o(\mathrm{tw}^2)} \cdot n^{\mathcal{O}(1)}$ under the ETH.

What's next?

There are many problems that I would like to solve...

In particular, several questions concerning $\mathcal{F}$-Deletion:

- Ultimate goal: classify the (asymptotically) tight complexity of $\mathcal{F}$-Deletion for every family $\mathcal{F}$... we are still very far from it.
- We do not even know if there exists some $\mathcal{F}$ such that $\mathcal{F}$-Deletion cannot be solved in time $2^{o(\mathrm{tw}^2)} \cdot n^{\mathcal{O}(1)}$ under the ETH.
- Only "missing" connected graph on at most 5 vertices: K_5.
  We think that $\{K_5\}$-Deletion is solvable in time $2^{\Theta(\mathrm{tw} \cdot \log \mathrm{tw})} \cdot n^{\mathcal{O}(1)}$.

What's next?

There are many problems that I would like to solve...

In particular, several questions concerning $\mathcal{F}$-Deletion:

- Ultimate goal: classify the (asymptotically) tight complexity of $\mathcal{F}$-Deletion for every family $\mathcal{F}$... we are still very far from it.
- We do not even know if there exists some $\mathcal{F}$ such that $\mathcal{F}$-Deletion cannot be solved in time $2^{o(\mathsf{tw}^2)} \cdot n^{\mathcal{O}(1)}$ under the ETH.
- Only "missing" connected graph on at most 5 vertices: K_5. We think that $\{K_5\}$-Deletion is solvable in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$.
- Conjecture For every connected planar graph H with $|V(H)| \geq 6$, $\mathcal{F}$-Deletion is solvable in time $2^{\Theta(\mathsf{tw} \cdot \log \mathsf{tw})} \cdot n^{\mathcal{O}(1)}$ under the ETH.

Gràcies!

LLIBERTAT
PRESOS POLÍTICS