Logics and Algorithms for Graph Minors

Giannos Stamoulis

AlGCo team

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

Committee:

Anuj Dawar	reviewer
Marcin Pilipczuk	reviewer
Pierre Fraigniaud	examiner
Frédéric Havet	examiner
Eun Jung Kim	examiner
Ignasi Sau	supervisor
Dimitrios M. Thilikos	co-supervisor

Amphithéâtre Jean Jacques Moreau, 12/12/2023

Computation as a mathematical subject

Computation as a mathematical subject

Study of automated computation by means of abstraction

Computation as a mathematical subject

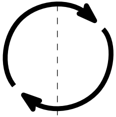
Study of automated computation by means of abstraction

What makes a computational problem
inherently difficult?

Computation as a mathematical subject

Study of automated computation by means of abstraction

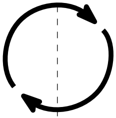
What makes a computational problem
inherently difficult?



Computation as a mathematical subject

Study of automated computation by means of abstraction

What makes a computational problem inherently difficult?

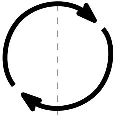


When can we have efficient algorithms?

Computation as a mathematical subject

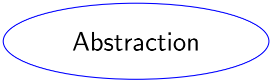
Study of automated computation by means of abstraction

What makes a computational problem inherently difficult?

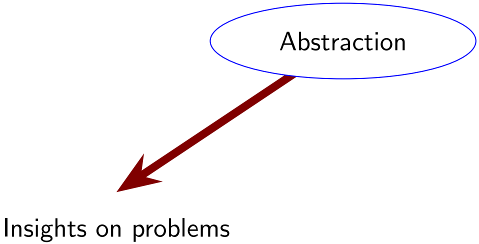


When can we have efficient algorithms?

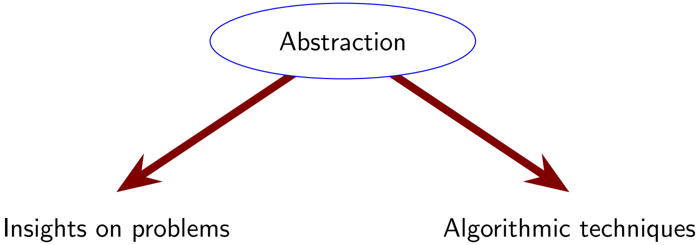
The power of abstraction



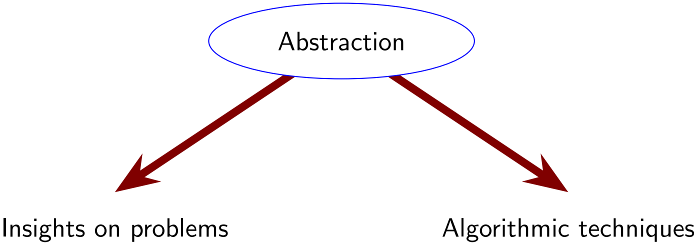
The power of abstraction



The power of abstraction

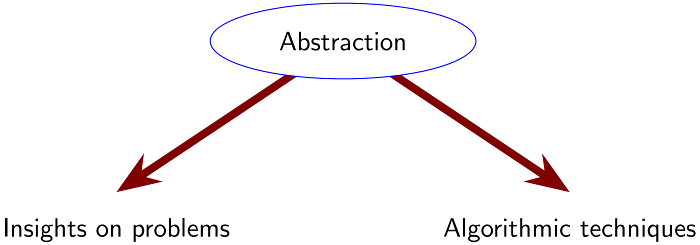


The power of abstraction



Common descriptive ground?

The power of abstraction



Common descriptive ground?

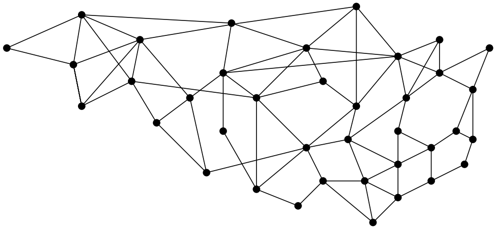
What is their potential?

Graphs and algorithms

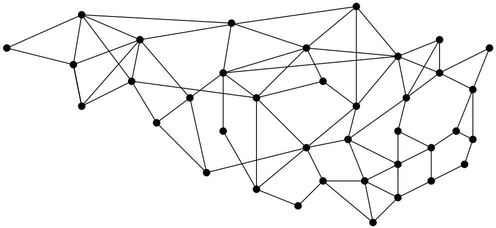
- Model of abstraction: *Graphs*

Graphs and algorithms

- Model of abstraction: *Graphs*



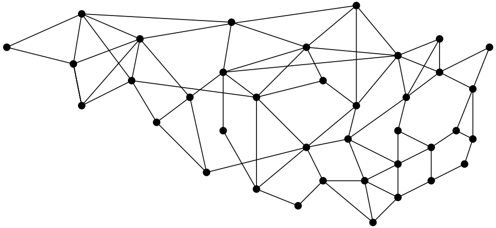
Graphs and algorithms



- Model of abstraction: *Graphs*

- *Decision problems*: answered by **YES** or **NO**

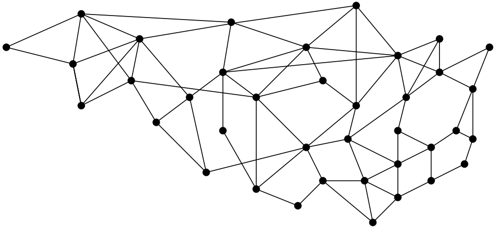
Graphs and algorithms



- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property $\texttt{X}$?

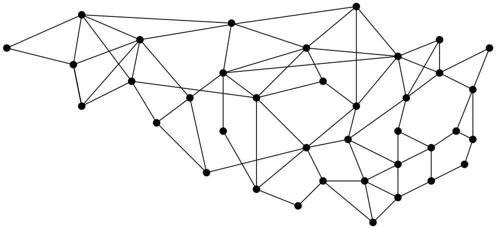
Graphs and algorithms



- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property X ?

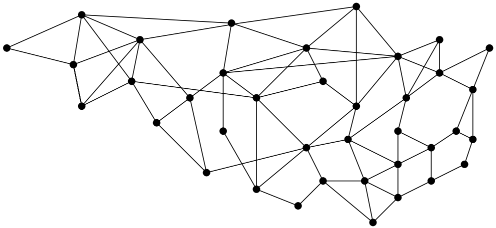
Graphs and algorithms



- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property $\texttt{X}$?

Graphs and algorithms

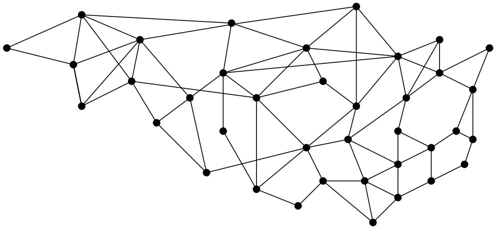


- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property X ?

▷ How to describe a property?

Graphs and algorithms and logic

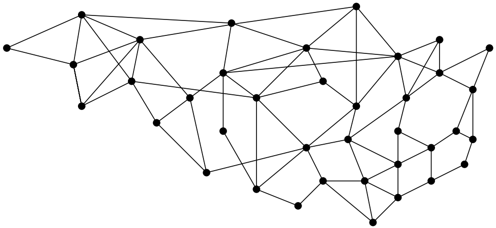


- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property $\texttt{X}$?

▷ How to describe a property? → Machine description

Graphs and algorithms and logic

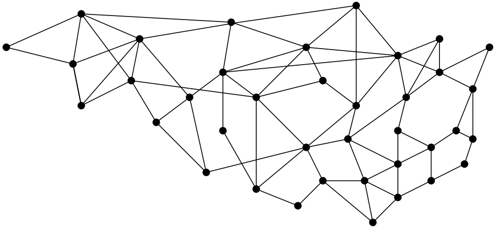


- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property X ?

▷ How to describe a property? → Machine description
→ Logic

Graphs and algorithms and logic



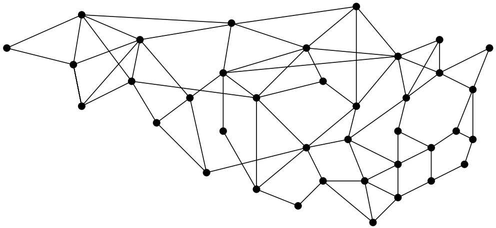
- Model of abstraction: *Graphs*

- *Decision problems*: answered by **YES** or **NO**

Given a graph G, does it have property `X` ?

▷ How to describe a property? → Machine description
→ Logic (*abstract language to describe properties/problems*)

Graphs and algorithms and logic



- Model of abstraction: *Graphs*
- *Decision problems*: answered by **YES** or **NO**

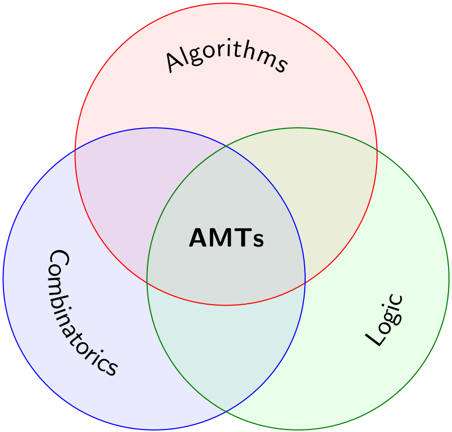
Given a graph G, does it have property X ?

▷ How to describe a property? → Machine description
→ Logic (*abstract language to describe properties/problems*)

▷ How to use the structure of the graph to obtain efficient algorithms ?

Meta-algorithmic perspective

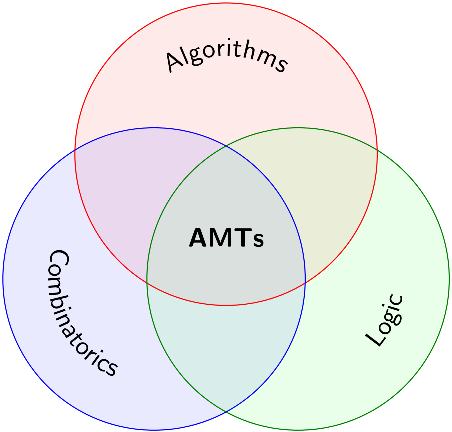
Algorithmic meta-theorems (**AMTs**):
General mathematical conditions that allow the automatic derivation of efficient algorithms.



Meta-algorithmic perspective

Algorithmic meta-theorems (**AMTs**):
General mathematical conditions that allow the automatic derivation of efficient algorithms.

Conditions: logical ($\mathbf{C}_L$) & combinatorial ($\mathbf{C}_C$)

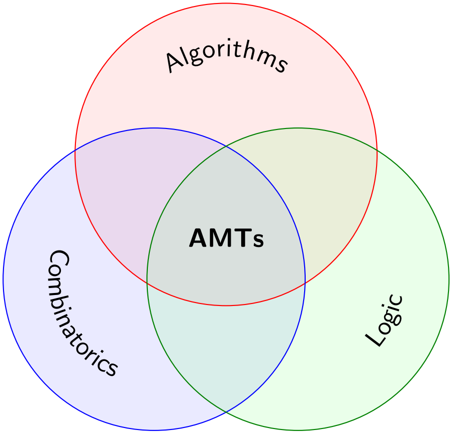


Meta-algorithmic perspective

Algorithmic meta-theorems (**AMTs**):
General mathematical conditions that allow the automatic derivation of efficient algorithms.

Conditions: logical ($\mathbf{C}_L$) & combinatorial ($\mathbf{C}_C$)

"every problem that is expressible by $\mathbf{C}_L$*,*
can be solved efficiently,
on instances restricted by $\mathbf{C}_C$*."*



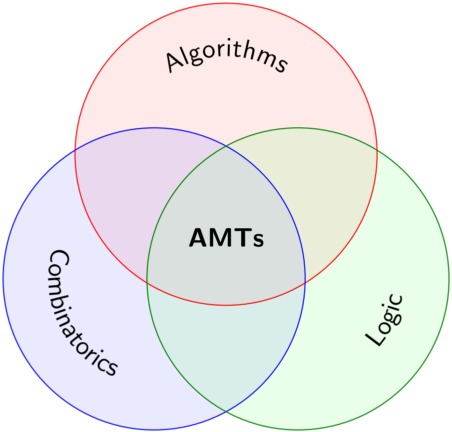
Meta-algorithmic perspective

Algorithmic meta-theorems (**AMTs**):
General mathematical conditions that allow the automatic derivation of efficient algorithms.

Conditions: logical ($\mathbf{C}_L$) & combinatorial ($\mathbf{C}_C$)

"every problem that is expressible by $\mathbf{C}_L$,
can be solved efficiently,
on instances restricted by $\mathbf{C}_C$."

"Algorithms that give algorithms"

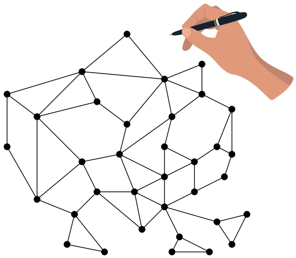


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

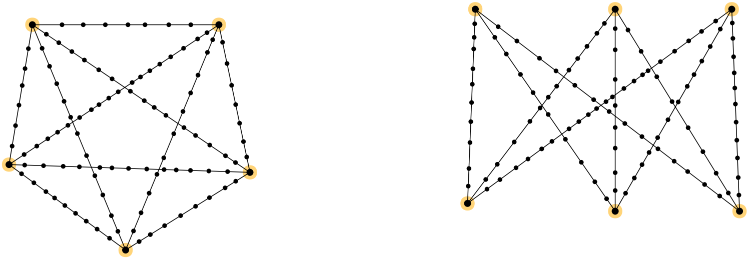


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain a subdivision of K_5 or $K_{3,3}$ as a subgraph.

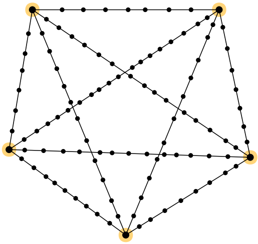


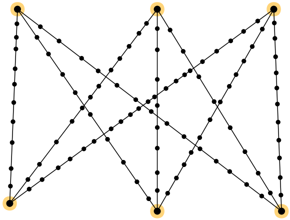
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



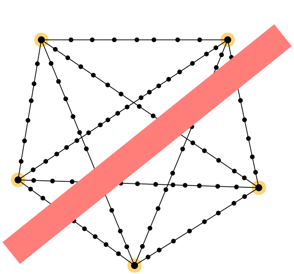


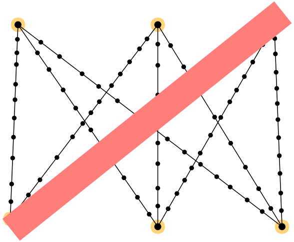
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



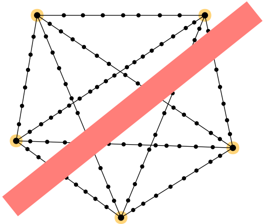


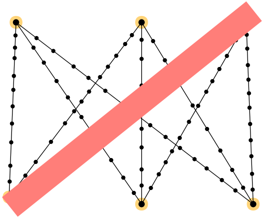
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



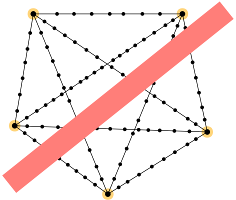


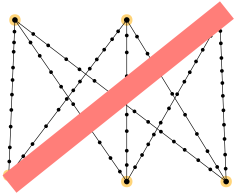
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



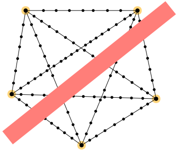


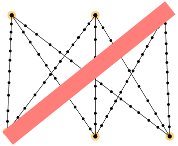
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



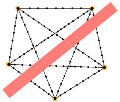


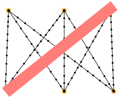
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



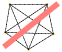


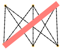
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



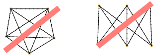


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

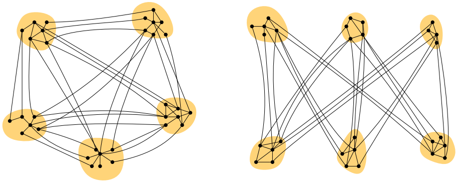
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.

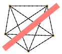


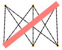
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

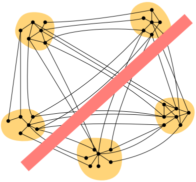
G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.

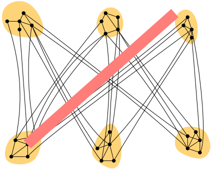




Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.



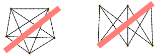


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

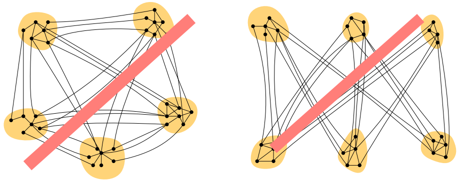
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.

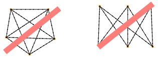


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

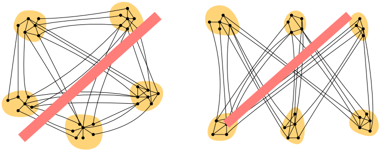
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.

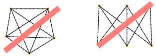


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

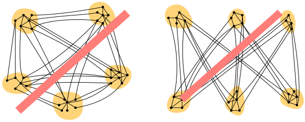
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.

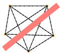


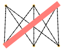
How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

Kuratowski-Pontryagin theorem (1930):

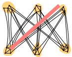
G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.





Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.

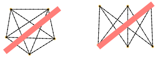


How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

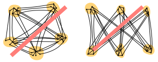
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.



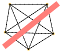
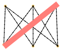
Erdős' conjecture:
Surface embeddability of graphs is characterized by a few obstructions.

How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

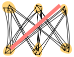
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.

Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.

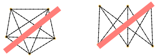
Erdős' conjecture:
Surface embeddability of graphs is characterized by a few obstructions.
(*minor-minimal* graphs not satisfying $\mathcal{P}$)

How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

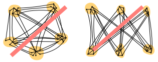
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.



Erdős' conjecture:
Surface embeddability of graphs is characterized by a few obstructions.
(*minor-minimal* graphs not satisfying $\mathcal{P}$)

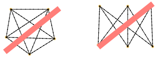
"Wagner's" conjecture:
Every minor-closed property is characterized by a few obstructions.

How to describe a property (without Logic)

When can a graph be drawn on the plane without crossings?

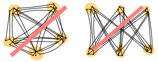
Kuratowski-Pontryagin theorem (1930):

G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a topological minor.



Wagner's theorem (1937):

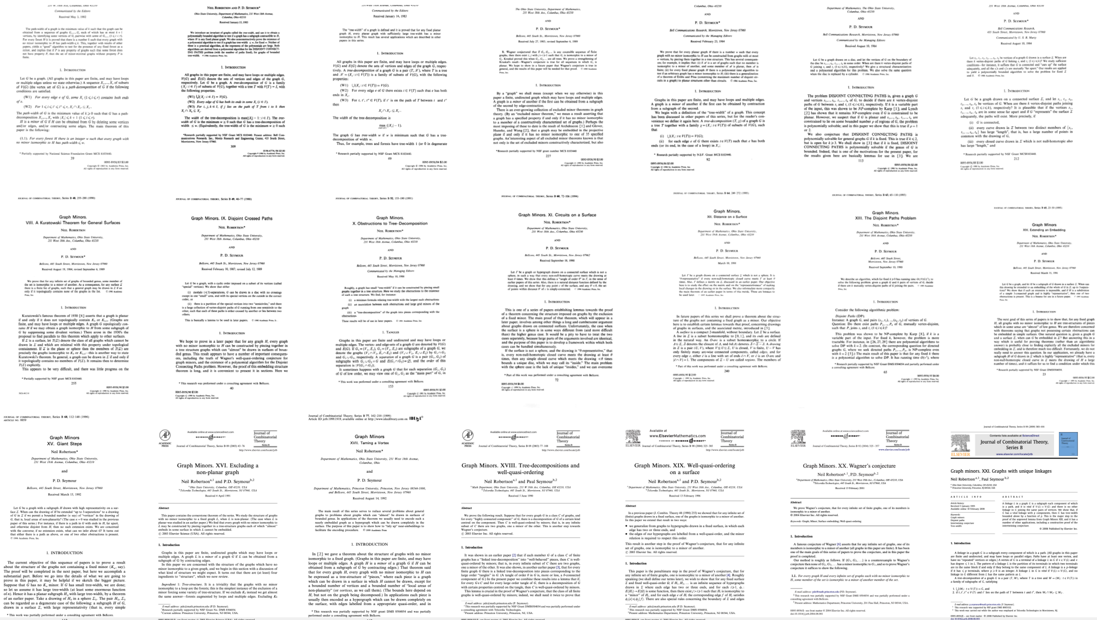
G is planar $\iff$ G does not contain K_5 or $K_{3,3}$ as a minor.



Erdős' conjecture:
Surface embeddability of graphs is characterized by a few obstructions.
(*minor-minimal* graphs not satisfying $\mathcal{P}$)

"Wagner's" conjecture:
Every minor-closed property is characterized by a few obstructions.
(*maintained on minors*)

Graph Minors series


Graph Minors. VIII. A Kuratowski Theorem for General Surfaces

Graph Minors. IX. Disjoint Crossed Paths

Graph Minors. X. Obstructions to Tree-Decomposition

Graph Minors. XI. Circuits on a Surface

Graph Minors. XII. Distance on a Surface

Graph Minors. XIII. The Disjoint Paths Problem

Graph Minors. XIV. Extending an Embedding

Graph Minors. XV. Giant Steps

Graph Minors. XVI. Excluding a non-planar graph

Graph Minors. XVII. Taming a Vortex

Graph Minors. XVIII. Tree-decompositions and well-quasi-ordering

Graph Minors. XIX. Well-quasi-ordering on a surface

Graph Minors. XX. Wagner's conjecture

Graph minors. XXI. Graphs with unique linkages

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

1) *Every minor-closed property is characterized by a few obstructions.*

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

1) *Every minor-closed property is characterized by a few obstructions.*

2) *Testing whether H is a minor of G can be done in polynomial time.*

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

1) *Every minor-closed property is characterized by a few obstructions.*

2) *Testing whether H is a minor of G can be done in polynomial time.*

Main algorithmic consequence of Graph Minors:
Every minor-closed property can be decided in polynomial time.

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

1) *Every minor-closed property is characterized by a few obstructions.*

2) *Testing whether H is a minor of G can be done in polynomial time.*

Main algorithmic consequence of Graph Minors:
Every minor-closed property can be decided in polynomial time.

- Deciding a minor-closed property is reduced to minor testing!

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

1) *Every minor-closed property is characterized by a few obstructions.*

2) *Testing whether H is a minor of G can be done in polynomial time.*

Main algorithmic consequence of Graph Minors:
Every minor-closed property can be decided in polynomial time.

- Deciding a minor-closed property is reduced to minor testing!

Example: *Planarity*. Can G be drawn on the plane without crossings?
In other words: Does G contain K_5 or $K_{3,3}$ as a minor?

Graph Minors series

A series of 23 papers by Robertson & Seymour [GM I, 1982],...,[GM XXIII, 2010].

Seminal results of the Graph Minors series:

1) *Every minor-closed property is characterized by a few obstructions.*

2) *Testing whether H is a minor of G can be done in polynomial time.*

Main algorithmic consequence of Graph Minors:
Every minor-closed property can be decided in polynomial time.

- Deciding a minor-closed property is reduced to minor testing!

Example: *Planarity*. Can G be drawn on the plane without crossings?
In other words: Does G contain K_5 or $K_{3,3}$ as a minor?

- The proof of 1) is non-constructive (does not give the obstructions) and is not expected to be constructive (in general).

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

$\downarrow$

parameter (k = value of the parameter)

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

$\downarrow$

parameter (k = value of the parameter)

Efficiency demand:

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

$\downarrow$

parameter (k = value of the parameter)

Efficiency demand:
Fixed-Parameter Tractable algorithms

Running time: $f(k) \cdot n^c$

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

$\downarrow$

parameter (k = value of the parameter)

Efficiency demand:
Fixed-Parameter Tractable algorithms

Running time: $\mathcal{O}_k(n^c)$

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

$\downarrow$

parameter (k = value of the parameter)

Efficiency demand:
Fixed-Parameter Tractable algorithms

Running time: $\mathcal{O}_k(n^c)$

▷ Vibrant branch of TCS & Mathematics the last ~30 years.

Parameterized viewpoint

- Graph Minors: structure $\rightarrow$ algorithms

Parameterized Computation (branch of TCS & Mathematics):
Study of auxiliary measure *conditioning* the computational complexity of problems.

$\downarrow$

parameter (k = value of the parameter)

Efficiency demand:
Fixed-Parameter Tractable algorithms

Running time: $\mathcal{O}_k(n^c)$

▷ Vibrant branch of TCS & Mathematics the last ∼30 years.

`Dream`: Meta-algorithmic viewpoint on Parameterized Computation.

General Goals

▷ When can we construct the obstruction set of a minor-closed property?

General Goals

▷ When can we construct the obstruction set of a minor-closed property?

General Goals

- When can we construct ~~the obstruction set of~~ a minor-closed property?
  a meta-algorithm deciding

General Goals

- When can we construct ~~the obstruction set of~~ a minor-closed property?
  a meta-algorithm deciding

- Algorithmic Graph Minors theory?

General Goals

- When can we construct ~~the obstruction set of~~ a minor-closed property?
  a meta-algorithm deciding

- Algorithmic Graph Minors theory?

Main objective of the thesis:

- Explore the meta-algorithmic potential of structural results of Graph Minors

General Goals

- When can we construct ~~the obstruction set of~~ a minor-closed property?
  a meta-algorithm deciding
- Algorithmic Graph Minors theory?

Main objective of the thesis:

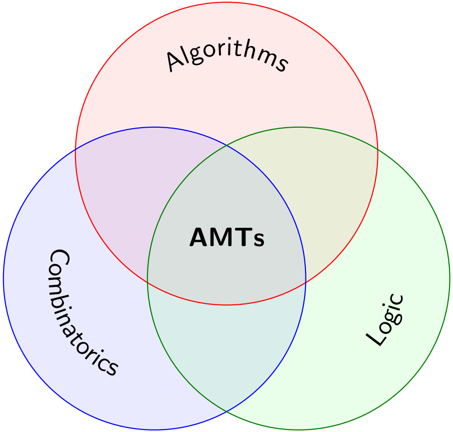
- Explore the meta-algorithmic potential of structural results of Graph Minors

Our contribution:

- A unified meta-algorithmic framework on minor exclusion.
- Extension to classes excluding topological minors.

Logics and Algorithms for Graph Minors

The 3 components of AMTs

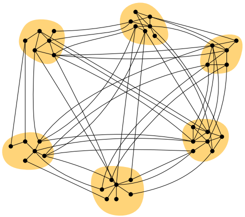


Flat wall theorem (*Local Structure theorem*) [GM XIII]

Given a graph G and two integers h, k, one of the following holds:

Flat wall theorem (*Local Structure theorem*) [GM XIII]

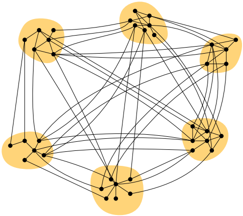
Given a graph G and two integers h, k, one of the following holds:



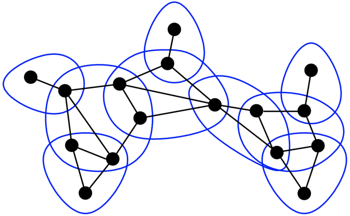
K_h is a minor of G,

Flat wall theorem (*Local Structure theorem*) [GM XIII]

Given a graph G and two integers h, k, one of the following holds:



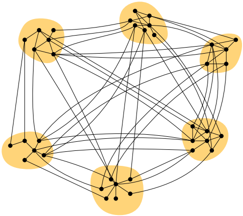
K_h is a minor of G,



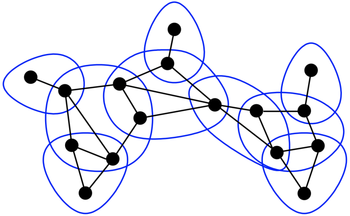
G has "small" treewidth (depending only on k and h), or

Flat wall theorem (*Local Structure theorem*) [GM XIII]

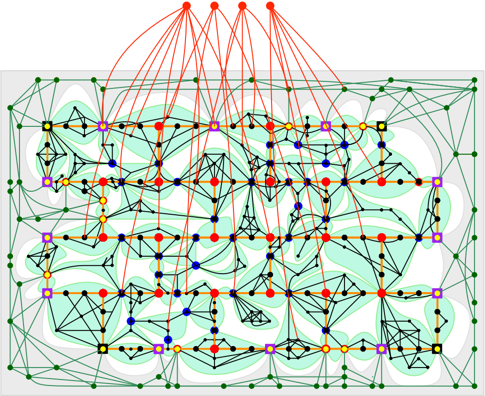
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



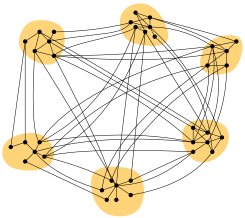
G has "small" treewidth (depending only on k and h), or



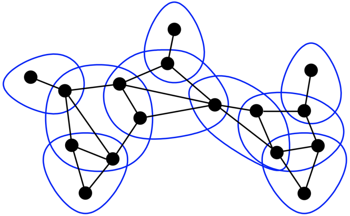
there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

Flat wall theorem (*Local Structure theorem*) [GM XIII]

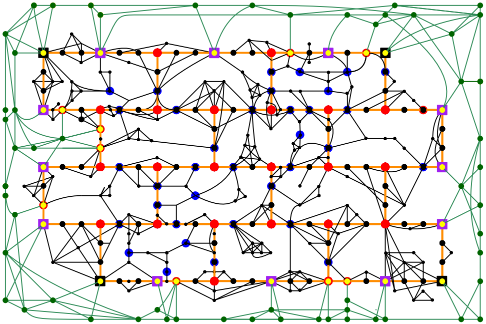
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



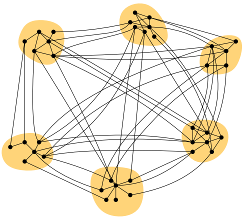
G has "small" treewidth (depending only on k and h), or



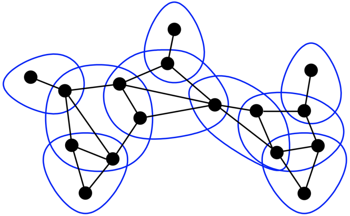
there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

Flat wall theorem (*Local Structure theorem*) [GM XIII]

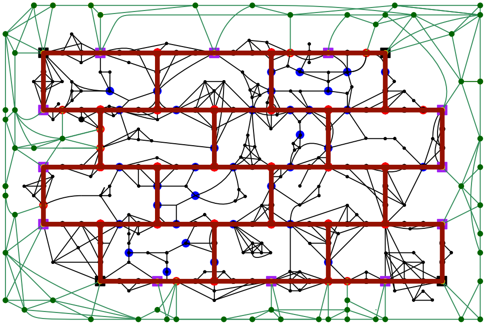
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



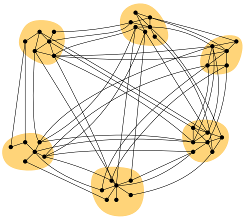
G has "small" treewidth (depending only on k and h), or



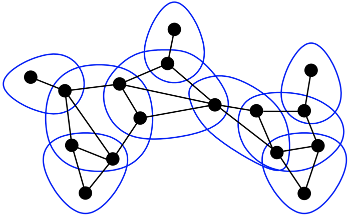
there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

Flat wall theorem (*Local Structure theorem*) [GM XIII]

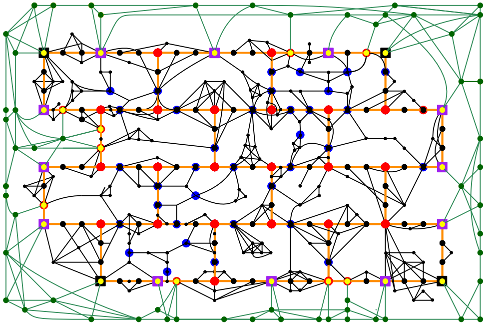
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



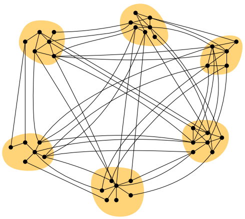
G has "small" treewidth (depending only on k and h), or



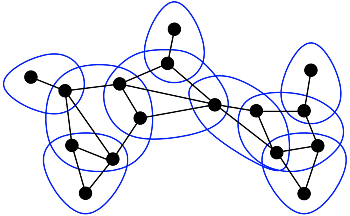
there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

Flat wall theorem (*Local Structure theorem*) [GM XIII]

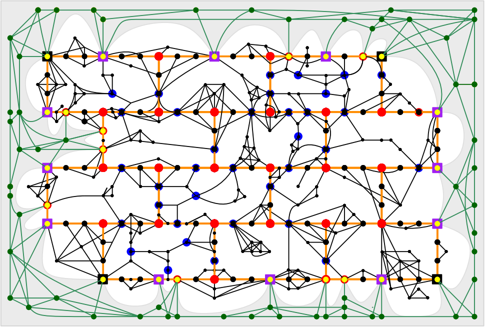
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



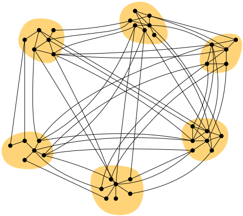
G has "small" treewidth (depending only on k and h), or



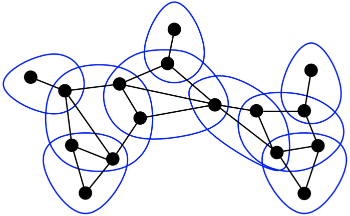
there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

Flat wall theorem (*Local Structure theorem*) [GM XIII]

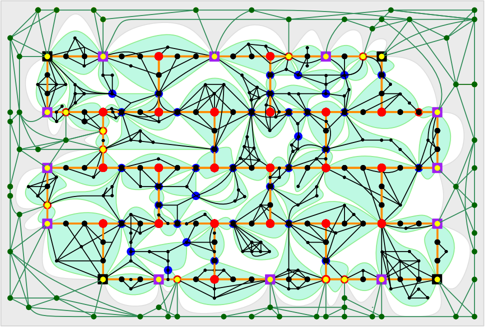
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



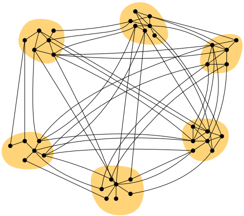
G has “small” treewidth (depending only on k and h), or



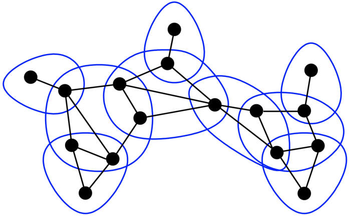
there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

Flat wall theorem (*Local Structure theorem*) [GM XIII]

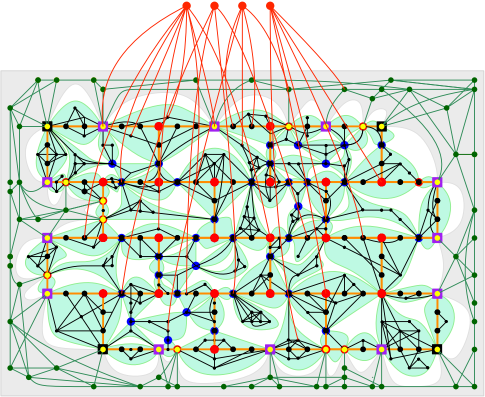
Given a graph G and two integers h, k, one of the following holds:



K_h is a minor of G,



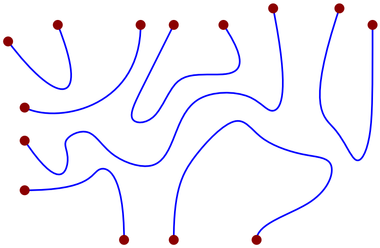
G has "small" treewidth (depending only on k and h), or



there is a set A of $f(h)$ vertices of G, such that $G - A$ contains a flat k-wall W.

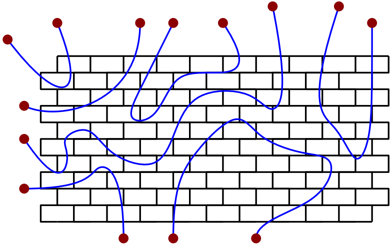
Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.



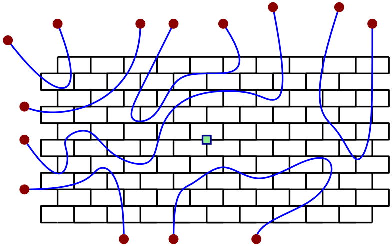
Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.



Irrelevant vertex technique [GM XIII, XXI, XXII]

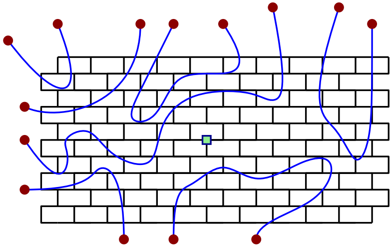
Introduced in [GM XIII] to solve the DISJOINT PATHS problem.



Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the DISJOINT PATHS problem.

- If instance is **simple** (has "small" treewidth), then problem is "easily" solvable.

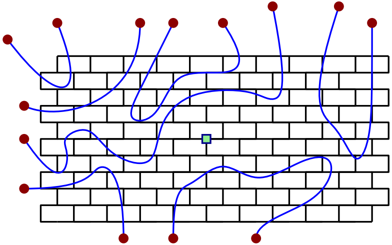


Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.

- If instance is **simple** (has "small" treewidth), then problem is "easily" solvable.
- If instance is **not simple enough** (has "large" treewidth), then get **simpler & equivalent** instance.

(by finding and removing irrelevant vertices)



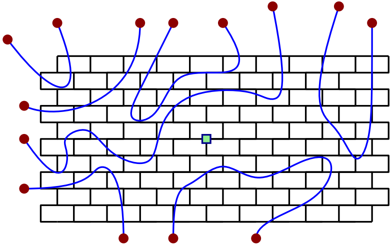
Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.

- If instance is **simple** (has "small" treewidth), then problem is "easily" solvable.
- If instance is **not simple enough** (has "large" treewidth), then get **simpler & equivalent** instance.

(by finding and removing irrelevant vertices)

▷ More than 50 papers using this technique.



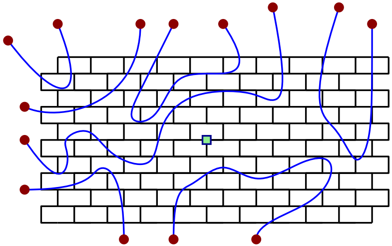
Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.

- If instance is **simple** (has "small" treewidth), then problem is "easily" solvable.
- If instance is **not simple enough** (has "large" treewidth), then get **simpler & equivalent** instance.

(by finding and removing irrelevant vertices)

▷ More than 50 papers using this technique.



▷ Why irrelevant vertices are *irrelevant*?

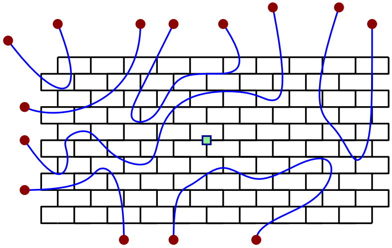
Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.

- If instance is **simple** (has "small" treewidth), then problem is "easily" solvable.
- If instance is **not simple enough** (has "large" treewidth), then get **simpler & equivalent** instance.

(by finding and removing irrelevant vertices)

▷ More than 50 papers using this technique.



▷ Why irrelevant vertices are *irrelevant*? *Unique Linkage theorem*

[GM XXI–XXII] [Adler, Kolliopoulos, Krause, Lokshtanov, Saurabh, Thilikos, 2017] [Kawarabayashi & Wollan, 2010] [Mazoit, 2013]

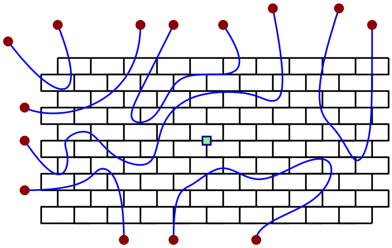
Irrelevant vertex technique [GM XIII, XXI, XXII]

Introduced in [GM XIII] to solve the Disjoint Paths problem.

- If instance is **simple** (has “small” treewidth), then problem is “easily” solvable.
- If instance is **not simple enough** (has “large” treewidth), then get **simpler & equivalent** instance.

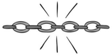
(by finding and removing irrelevant vertices)

▷ More than 50 papers using this technique.



▷ Why irrelevant vertices are *irrelevant*? *Unique Linkage theorem*

[GM XXI–XXII] [Adler, Kolliopoulos, Krause, Lokshtanov, Saurabh, Thilikos, 2017] [Kawarabayashi & Wollan, 2010] [Mazoit, 2013]

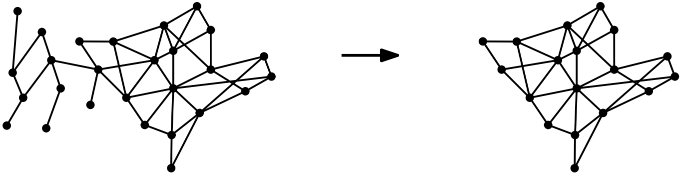
Flat Wall theorem  Irrelevant vertex technique

The algorithmic paradigm of Simplification

Irrelevant vertex technique describes a simplification procedure (a **data reduction**).

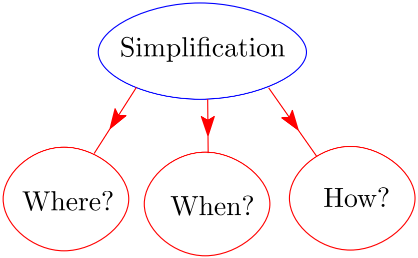
General question: *"How to simplify the input?"*

Example: Does G contain a cycle of length 5?



Designing algorithms using Simplification

- ▷ How Simplification can aid to the design of algorithms?
- In simplified instances, problems are solved more easily.



- We need abstraction and deep understanding of the irrelevant vertex technique.

Meta-algorithmization of the irrelevant vertex technique

Our viewpoint:
Irrelevant vertex technique = instantiation of the **algorithmic paradigm of** Simplification.

Meta-algorithmization of the irrelevant vertex technique

Our viewpoint:
Irrelevant vertex technique = instantiation of the **algorithmic paradigm of** Simplification.

▷ How general this technique can be ? Meta-algorithmics of Graph Minors?

Meta-algorithmization of the irrelevant vertex technique

Our viewpoint:
Irrelevant vertex technique = instantiation of the **algorithmic paradigm of** Simplification.

- ▷ How general this technique can be ? Meta-algorithmics of Graph Minors?
- ▷ What problems can we solve, when excluding a (topological) minor?

Meta-algorithmization of the irrelevant vertex technique

Our viewpoint:
Irrelevant vertex technique = instantiation of the **algorithmic paradigm of** $\mathrm{Simplification}$.

- ▷ How general this technique can be ? Meta-algorithmics of Graph Minors?
- ▷ What problems can we solve, when excluding a (topological) minor?
- ▷ What properties can we deal with?

Meta-algorithmization of the irrelevant vertex technique

Our viewpoint:
Irrelevant vertex technique = instantiation of the **algorithmic paradigm of** Simplification.

- How general this technique can be ? Meta-algorithmics of Graph Minors?
- What problems can we solve, when excluding a (topological) minor?
- What properties can we deal with?

We resort to Logic.

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),
  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),
  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.
  Question: G has the property described by φ?

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),

  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.

  Question: ~~G has the property described by φ~~?

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),

  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.

  Question: ~~G has the property described by φ~~?

  G satisfies φ? Written as "$G \models \varphi$?"

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),

  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.

  Question: ~~G has the property described by φ~~?

  G satisfies φ? Written as "$G \models \varphi$?"

AMTs in terms of model checking:

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),
  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.
  Question: ~~G has the property described by φ~~?
  G satisfies φ? Written as "$G \models \varphi$?"

AMTs in terms of model checking:
Given logic $\mathcal{L}$ and graph class $\mathcal{C}$,

Model checking for $\mathcal{L}$ can be solved in polynomial time on graphs from $\mathcal{C}$.

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),
  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.
  Question: ~~G has the property described by φ~~?
  G satisfies φ? Written as "$G \models \varphi$?"

AMTs in terms of model checking:
Given logic $\mathcal{L}$ and graph class $\mathcal{C}$,

Model checking for $\mathcal{L}$ can be solved in polynomial time on graphs from $\mathcal{C}$.

$$\mathcal{O}_{|\varphi|, c_{\mathcal{C}}}(n^c)$$

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),

Input: A formula $\varphi \in \mathcal{L}$ and a graph G.

Question: ~~G has the property described by φ~~?

G satisfies φ? Written as "$G \models \varphi$?"

AMTs in terms of model checking:

Given logic $\mathcal{L}$ and graph class $\mathcal{C}$,

Model checking for $\mathcal{L}$ can be solved in polynomial time on graphs from $\mathcal{C}$.

$$\mathcal{O}_{|\varphi|, c_{\mathcal{C}}}(n^c)$$

size of input graph

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),

  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.

Question: ~~G has the property described by φ~~?

G satisfies φ? Written as "$G \models \varphi$?"

AMTs in terms of model checking:

Given logic $\mathcal{L}$ and graph class $\mathcal{C}$,

Model checking for $\mathcal{L}$ can be solved in polynomial time on graphs from $\mathcal{C}$.

$$\mathcal{O}_{|\varphi|, c_{\mathcal{C}}}(n^c)$$

size of input formula

size of input graph

Model checking problem for a logic $\mathcal{L}$

- Given a logic $\mathcal{L}$ (on the vocabulary of graphs),
  Input: A formula $\varphi \in \mathcal{L}$ and a graph G.
  Question: ~~G has the property described by φ~~?
  G satisfies φ? Written as "$G \models \varphi$?"

AMTs in terms of model checking:
Given logic $\mathcal{L}$ and graph class $\mathcal{C}$,

Model checking for $\mathcal{L}$ can be solved in polynomial time on graphs from $\mathcal{C}$.

$$\mathcal{O}_{|\varphi|, c_{\mathcal{C}}}(n^{c})$$

size of input formula

constants depending on $\mathcal{C}$

size of input graph

First-Order and Monadic Second-Order logic

First-**O**rder logic **(FO)**:

First-Order and Monadic Second-Order logic

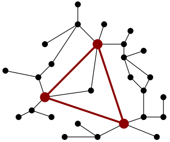
First-**O**rder logic **(FO)**:

$x = y \mid \textbf{adj}(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$

First-Order and Monadic Second-Order logic

First-**O**rder logic **(FO)**:

$x = y \mid \mathbf{adj}(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$

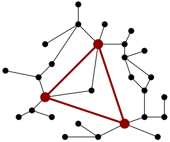


- Does G contain H as a subgraph? $\exists x \exists y \exists z \left(\mathbf{adj}(x, y) \wedge \mathbf{adj}(y, z) \wedge \mathbf{adj}(x, z)\right)$

First-Order and Monadic Second-Order logic

First-**O**rder logic **(FO)**:

$x = y \mid \mathbf{adj}(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$



- Does G contain H as a subgraph? $\exists x \exists y \exists z \left(\mathbf{adj}(x, y) \wedge \mathbf{adj}(y, z) \wedge \mathbf{adj}(x, z) \right)$

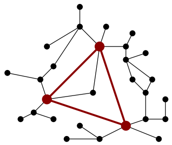
Monadic **S**econd-**O**rder logic (**MSO**):

$x = y \mid \mathbf{adj}(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi \mid \forall X \varphi \mid \exists X \varphi$

First-Order and Monadic Second-Order logic

First-**O**rder logic **(FO)**:

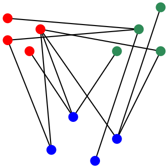
$x = y \mid \mathbf{adj}(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$



- Does G contain H as a subgraph? $\exists x \exists y \exists z \left(\mathbf{adj}(x, y) \wedge \mathbf{adj}(y, z) \wedge \mathbf{adj}(x, z)\right)$

Monadic **S**econd-**O**rder logic **(MSO)**:

$x = y \mid \mathbf{adj}(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi \mid \forall X \varphi \mid \exists X \varphi$



- Is G 3-*colorable*?

$$\exists V_1 \exists V_2 \exists V_3 \Big(\big(\forall x \ (x \in V_1 \vee x \in V_2 \vee x \in V_3)\big) \wedge \text{partition}(V_1, V_2, V_3)$$

$$\wedge \big(\forall x \forall y \ (x, y \in V_1) \vee (x, y \in V_2) \vee (x, y \in V_3) \implies \neg\mathbf{adj}(x, y)\big)\Big)$$

AMTs for FO and MSO

bounded treewidth [Courcelle,1990] [Arnborg, Lagergren, Seese, 1991] [Borie, Parker, Tovey, 1992]
bounded cliquewidth [Courcelle, Makowski, Rotics, 2000] [Oum & Seymour, 2006] **MSO**

---

bounded degree [Seese, 1996] **FO**
locally bounded treewidth [Frick & Grohe, 2001]
excluding a minor [Flum & Grohe, 2001]
locally excluding a minor [Dawar, Grohe, Kreutzer, 2007]
bounded expansion [Dvořák, Kráľ, Thomas, 2011]
nowhere dense [Grohe, Kreutzer, Siebertz, 2017]
bounded twinwidth [Bonnet, Kim, Thomassé, Watrigant, 2022]
structurally bounded degree [Gajarský, Hliněný, Lokshtanov, Obdržálek, Ramanujan, 2016]
structurally bounded expansion [Gajarský, Kreutzer, Nešetřil, Ossona de Mendez, Mi. Pilipczuk, Siebertz, Toruńczyk, 2018]
structurally nowhere dense [Dreier, Mählmann, Siebertz, 2023]
structurally bounded local cliquewidth [Bonnet, Dreier, Gajarský, Kreutzer, Mählmann, Simon, Toruńczyk, 2022]
monadically stable [Dreier, Eleftheriadis, Mählmann, McCarty, Mi. Pilipczuk, Toruńczyk, 2023]
monadically NIP/dependent ?

Algorithmic paradigms in form of AMTs

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

▷ AMTs for **MSO** = Meta-algorithmization of Dynamic Programming & Compositionality
based on tree-decomposability

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

▷ AMTs for **MSO** = Meta-algorithmization of Dynamic Programming & Compositionality
based on tree-decomposability

Commonly refered as *Courcelle's theorem*.

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

▷ AMTs for **MSO** = Meta-algorithmization of Dynamic Programming & Compositionality
based on tree-decomposability

Commonly refered as *Courcelle's theorem*.

- **Locality**: Focusing on "local" parts of the input is enough to solve the problem.

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

▷ AMTs for **MSO** = Meta-algorithmization of Dynamic Programming & Compositionality
based on tree-decomposability

Commonly refered as *Courcelle's theorem*.

- **Locality**: Focusing on "local" parts of the input is enough to solve the problem.
- **Separability**: Input can be split into well-separated parts.

Algorithmic paradigms in form of AMTs

- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

▷ AMTs for **MSO** = Meta-algorithmization of Dynamic Programming & Compositionality
based on tree-decomposability

Commonly refered as *Courcelle's theorem*.

- **Locality**: Focusing on "local" parts of the input is enough to solve the problem.
- **Separability**: Input can be split into well-separated parts.
- **Representative witnesses**.

Algorithmic paradigms in form of AMTs

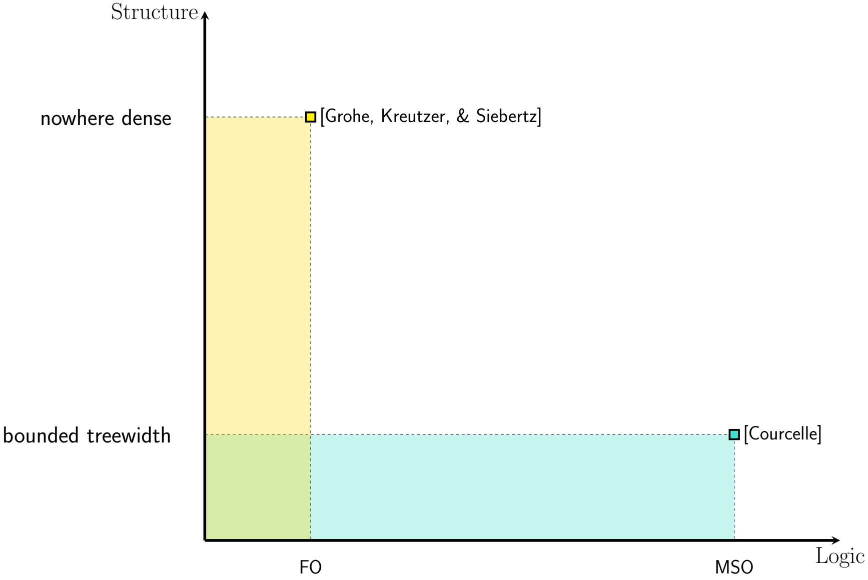
- **Dynamic Programming**: Recursive breaking into smaller subproblems.
- **Compositionality**: Combining solutions of subproblems.

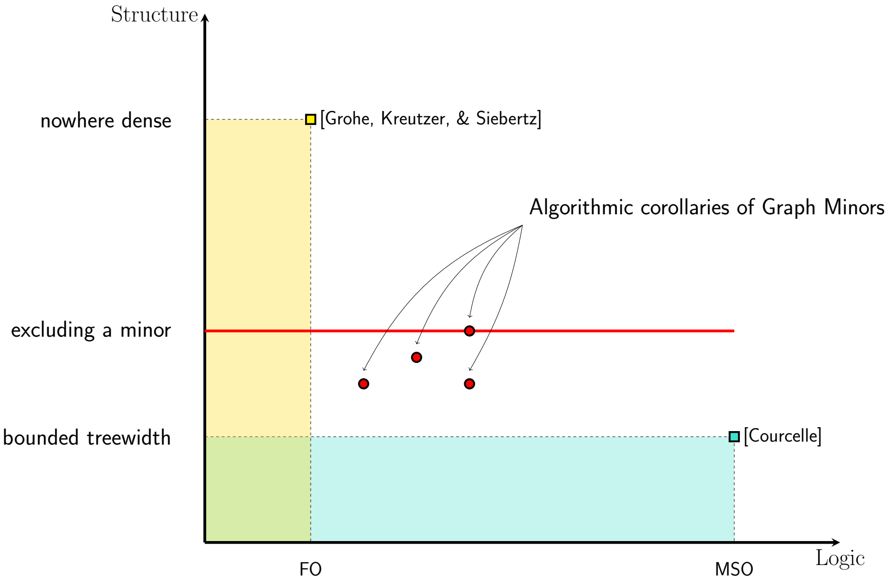
▷ AMTs for **MSO** = Meta-algorithmization of Dynamic Programming & Compositionality
based on tree-decomposability

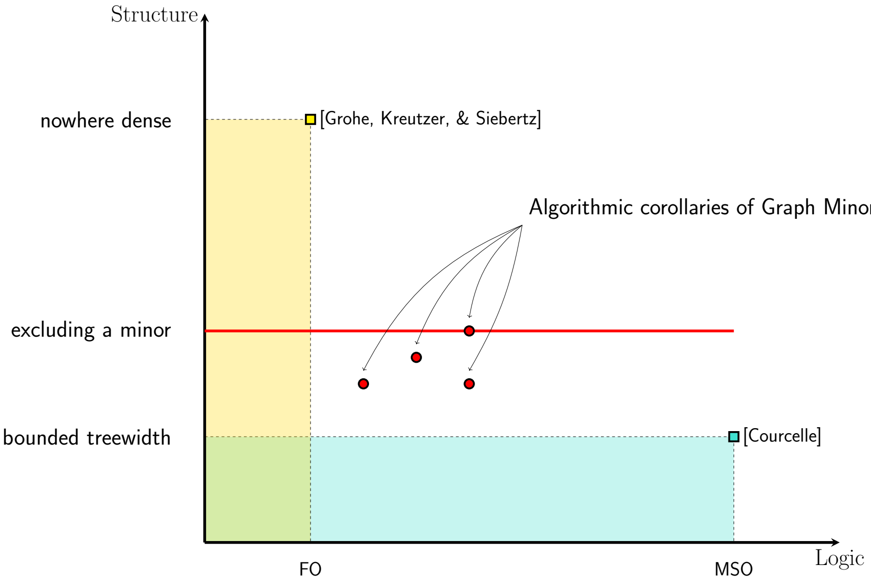
Commonly refered as *Courcelle's theorem*.

- **Locality**: Focusing on "local" parts of the input is enough to solve the problem.
- **Separability**: Input can be split into well-separated parts.
- **Representative witnesses**.

▷ AMTs for **FO** = Meta-algorithmization of Locality & Separability & Representative witnesses
based on sparsity


Structure
nowhere dense
[Grohe, Kreutzer, & Siebertz]
bounded treewidth
[Courcelle]
FO
MSO
Logic


Structure
nowhere dense
[Grohe, Kreutzer, & Siebertz]
Algorithmic corollaries of Graph Minors
excluding a minor
bounded treewidth
[Courcelle]
FO
MSO
Logic



We lack of a logical-based theory for Simplification.

Logics and Algorithms for Graph Minors

Algorithmic paradigm	*Logic*
Dynamic Programming / Compositionality	MSO
Locality / ...	FO
Simplification	?

Challenge: Find a logic encompassing the algorithmic paradigm of Simplification.

Logics and Algorithms for Graph Minors

Algorithmic paradigm	*Logic*
Dynamic Programming / Compositionality	MSO
Locality / ...	FO
Simplification	?

Challenge: Find a logic encompassing the algorithmic paradigm of Simplification.

- A meta-algorithmic theory of Graph Minors?

Logics and Algorithms for Graph Minors

Algorithmic paradigm	*Logic*
Dynamic Programming / Compositionality	MSO
Locality / ...	FO
Simplification	?

Challenge: Find a logic encompassing the algorithmic paradigm of Simplification.

- A meta-algorithmic theory of Graph Minors? Bidimensionality theory & Meta-kernelization

Logics and Algorithms for Graph Minors

Algorithmic paradigm	*Logic*
Dynamic Programming / Compositionality	MSO
Locality / ...	FO
Simplification	?

Challenge: Find a logic encompassing the algorithmic paradigm of Simplification.

- A meta-algorithmic theory of Graph Minors? Bidimensionality theory & Meta-kernelization

▷ We need to *create* a new combinatorial ground for such a theory.

Logics and Algorithms for Graph Minors

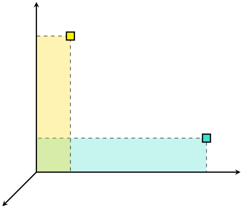
Algorithmic paradigm	*Logic*
Dynamic Programming / Compositionality	MSO
Locality / ...	FO
Simplification	?

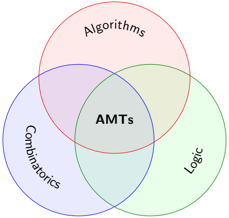
Challenge: Find a logic encompassing the algorithmic paradigm of Simplification.

- A meta-algorithmic theory of Graph Minors? Bidimensionality theory & Meta-kernelization

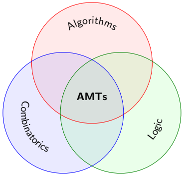
▷ We need to *create* a new combinatorial ground for such a theory.

▷ "**Efficiency dimension**" of AMTs?





Results



Combinatorial & Algorithmic tools

[Sau, *Stamoulis*, Thilikos. A more accurate view of the Flat Wall Theorem]
Under revision. Revised version in Journal of Graph Theory (**JGT**)

[Golovach, *Stamoulis*, Thilikos. Combing a Linkage in an Annulus]
*SIAM Journal on Discrete Mathematics (**SIDMA**), 2023*

AMTs

[Golovach, *Stamoulis*, Thilikos. Model-Checking for First-Order Logic with Disjoint Paths Predicates in Proper Minor-Closed Graph Classes]
SODA 2023

[Schirrmacher, Siebertz, *Stamoulis*, Thilikos, Vigny. Model Checking Disjoint-Paths Logic on Topological-Minor-Free Graph Classes]
Unpublished
[Fomin, Golovach, Sau, *Stamoulis*, Thilikos. Compound Logics for Modification Problems]
ICALP 2023

Efficiency dimension

[Sau, *Stamoulis*, Thilikos. k-apices of minor-closed graph classes. I. Bounding the obstructions]
*Journal of Combinatorial Theory, Series B (**JCTB**), 2023*

[Sau, *Stamoulis*, Thilikos. k-apices of minor-closed graph classes. II. Parameterized algorithms]
ICALP 2020 / ACM Transactions on Algorithms (**TALG**), 2022

[Morelle, Sau, *Stamoulis*, Thilikos. Faster parameterized algorithms for modification problems to minor-closed classes]
ICALP 2023

[Golovach, *Stamoulis*, Thilikos. Hitting Topological Minor Models in Planar Graphs is Fixed Parameter Tractable]
SODA 2020 / ACM Transactions on Algorithms (**TALG**), 2023

Combinatorial & algorithmic support of our AMTs

Enhanced algorithmic versions of the Flat Wall theorem

We build on the viewpoint of [Kawarabayashi, Thomas, Wollan, 2018].

Enhanced algorithmic versions of the Flat Wall theorem

We build on the viewpoint of [Kawarabayashi, Thomas, Wollan, 2018].

(Algorithmic enhancement of) Flat Wall theorem

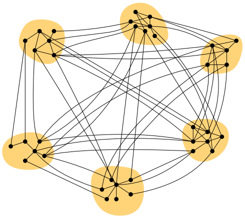
Input: graph G, integers r, t,
Output:

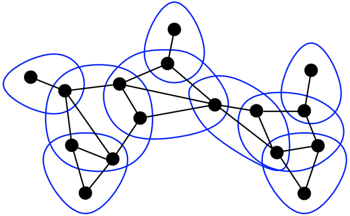
- either a report that K_t is a minor of G or G has treewidth $\mathcal{O}_t(r)$, or
- a set $A \subseteq V(G)$ of size poly(t) and a flat wall W of $G - A$ of height r, "whose perimeter crops a graph of treewidth $\mathcal{O}_t(r)$".

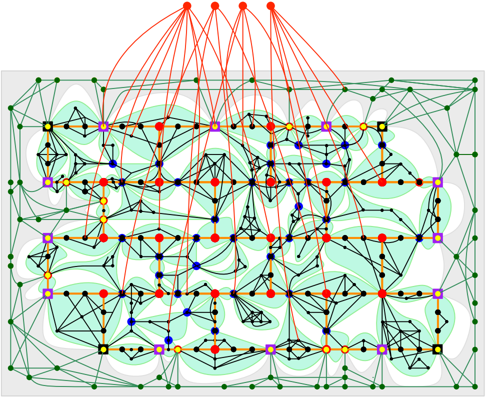
Running time: $2^{\mathcal{O}_t(r^2)} \cdot n$

Enhanced algorithmic versions of the Flat Wall theorem

We build on the viewpoint of [Kawarabayashi, Thomas, Wollan, 2018].

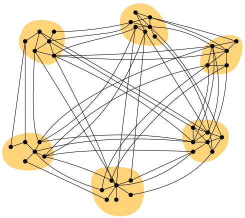


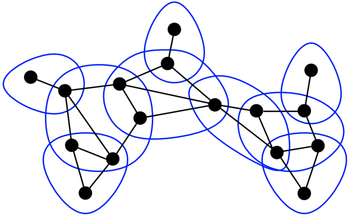


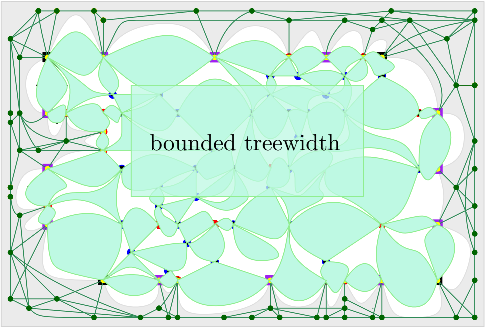


Enhanced algorithmic versions of the Flat Wall theorem

We build on the viewpoint of [Kawarabayashi, Thomas, Wollan, 2018].







Enhanced algorithmic versions of the Flat Wall theorem

We build on the viewpoint of [Kawarabayashi, Thomas, Wollan, 2018].

(Algorithmic enhancement of) Flat Wall theorem

Input: graph G, integers r, t,
Output:

- either a report that K_t is a minor of G or G has treewidth $\mathcal{O}_t(r)$, or
- a set $A \subseteq V(G)$ of size poly(t) and a flat wall W of $G - A$ of height r, "whose perimeter crops a graph of treewidth $\mathcal{O}_t(r)$".

Running time: $2^{\mathcal{O}_t(r^2)} \cdot n$

Enhanced algorithmic versions of the Flat Wall theorem

We build on the viewpoint of [Kawarabayashi, Thomas, Wollan, 2018].

(Algorithmic enhancement of) Flat Wall theorem

Input: graph G, integers r, t,
Output:

- either a report that K_t is a minor of G or G has treewidth $\mathcal{O}_t(r)$, or
- a set $A \subseteq V(G)$ of size poly(t) and a flat wall W of $G - A$ of height r, "whose perimeter crops a graph of treewidth $\mathcal{O}_t(r)$".

Running time: $2^{\mathcal{O}_t(r^2)} \cdot n$

- We introduce new combinatorial & algorithmic tools for flat walls, needed in our AMTs

Combing Linkages

How to deal with linkages?

Combing Linkages

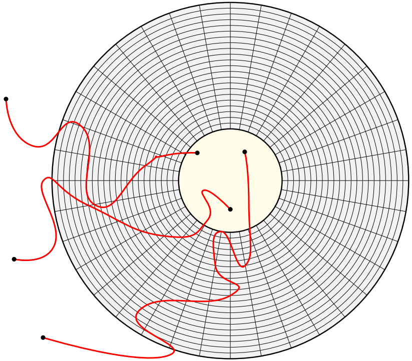
How to deal with linkages?

▷ Avoiding a vertex is not enough! **We need to comb!**

Combing Linkages

How to deal with linkages?

▷ Avoiding a vertex is not enough! **We need to comb!**



Combing Linkages

How to deal with linkages?

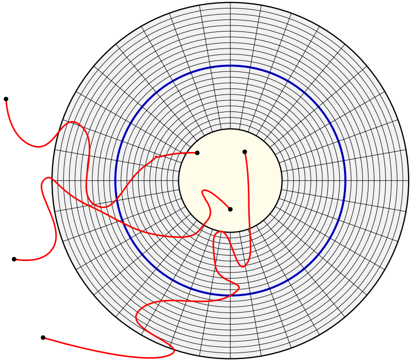
▷ Avoiding a vertex is not enough! **We need to comb!**

Linkage Combing Lemma

There is a function $f : \mathbb{N} \to \mathbb{N}$ such that if

- G is a partially disk-embedded graph,
- $(\mathcal{C}, \mathcal{P})$ is a disk-embedded railed annulus of size $f(k)$, and
- L is an annulus-avoiding linkage of size $\leqslant k$,

then there is an equivalent linkage L' that traverses the middle cycle of $\mathcal{C}$ through $\mathcal{P}$.



Combing Linkages

How to deal with linkages?

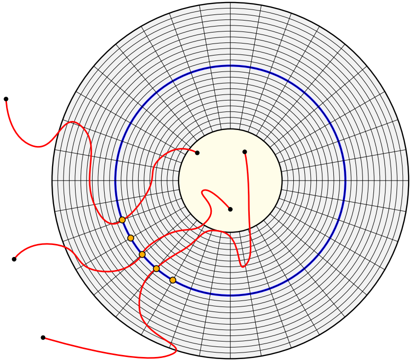
▷ Avoiding a vertex is not enough! **We need to comb!**

Linkage Combing Lemma

There is a function $f : \mathbb{N} \to \mathbb{N}$ such that if

- G is a partially disk-embedded graph,
- $(\mathcal{C}, \mathcal{P})$ is a disk-embedded railed annulus of size $f(k)$, and
- L is an annulus-avoiding linkage of size $\leqslant k$,

then there is an equivalent linkage L' that traverses the middle cycle of $\mathcal{C}$ through $\mathcal{P}$.



Combing Linkages

How to deal with linkages?

▷ Avoiding a vertex is not enough! **We need to comb!**

Linkage Combing Lemma

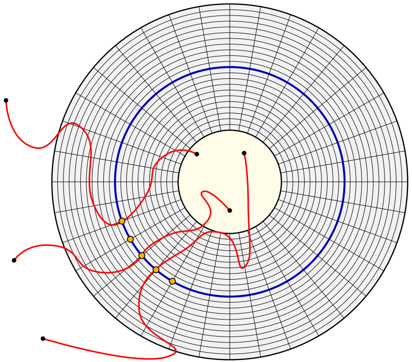
There is a function $f : \mathbb{N} \to \mathbb{N}$ such that if

- G is a partially disk-embedded graph,

- $(\mathcal{C}, \mathcal{P})$ is a disk-embedded railed annulus of size $f(k)$, and

- L is an annulus-avoiding linkage of size $\leqslant k$,

then there is an equivalent linkage L' that traverses the middle cycle of $\mathcal{C}$ through $\mathcal{P}$.



Strengthening of the *Unique Linkage theorem*.

Combing Linkages

How to deal with linkages?

▷ Avoiding a vertex is not enough! **We need to comb!**

Linkage Combing Lemma

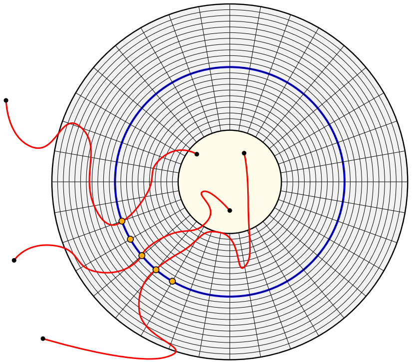
There is a function $f : \mathbb{N} \to \mathbb{N}$ such that if

- G is a partially disk-embedded graph,

- $(\mathcal{C}, \mathcal{P})$ is a disk-embedded railed annulus of size $f(k)$, and

- L is an annulus-avoiding linkage of size $\leqslant k$,

then there is an equivalent linkage L' that traverses the middle cycle of $\mathcal{C}$ through $\mathcal{P}$.



Strengthening of the *Unique Linkage theorem*.

Importance: *Finitely "represent" paths*

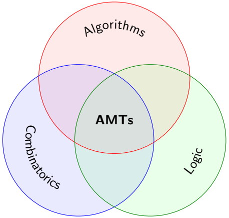
Recap of the combinatorial and algorithmic support

- Enhanced algorithmic versions of the Flat Wall theorem.

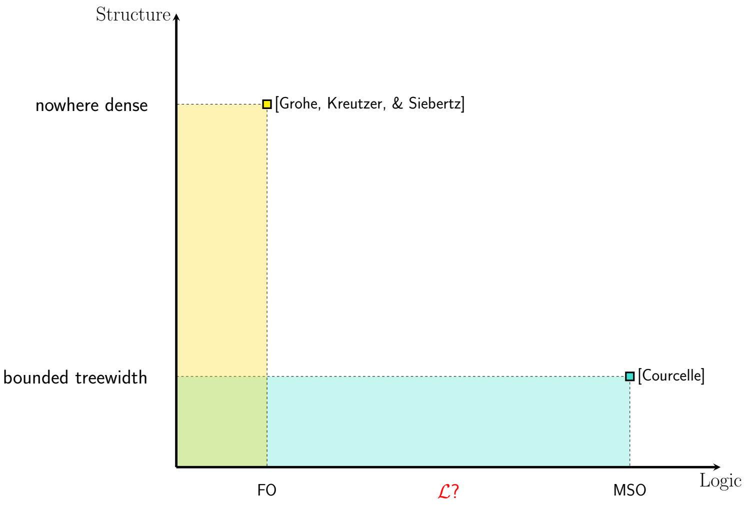
[Sau, *Stamoulis*, & Thilikos, A more accurate view of the Flat Wall Theorem]
Under revision. Revised version in *Journal of Graph Theory* (**JGT**)

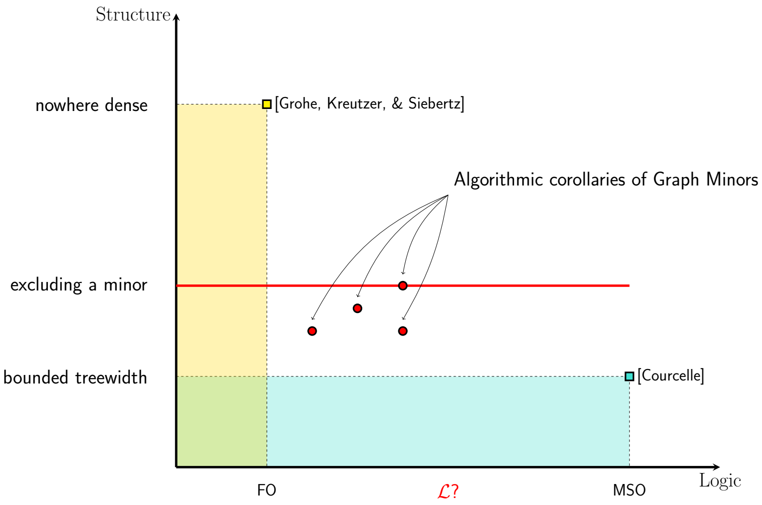
- Combing linkages in annuli.

[Golovach, *Stamoulis*, & Thilikos, Combing a Linkage in an Annulus]
SIAM Journal on Discrete Mathematics (**SIDMA**), 2023

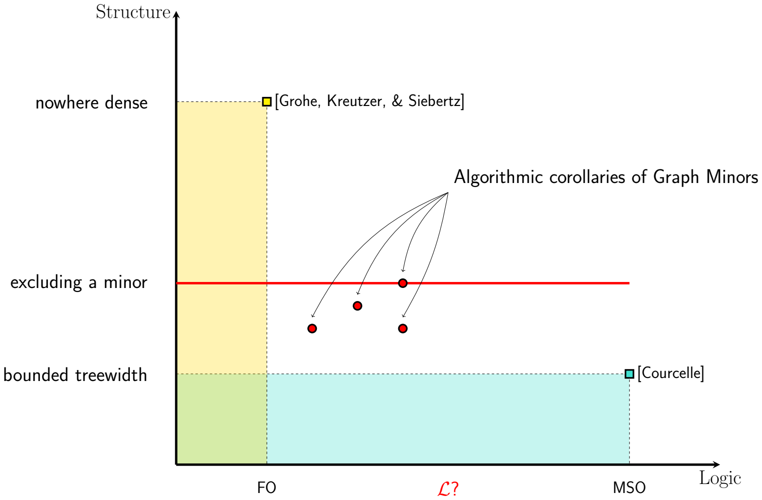


Our Algorithmic Meta-Theorems

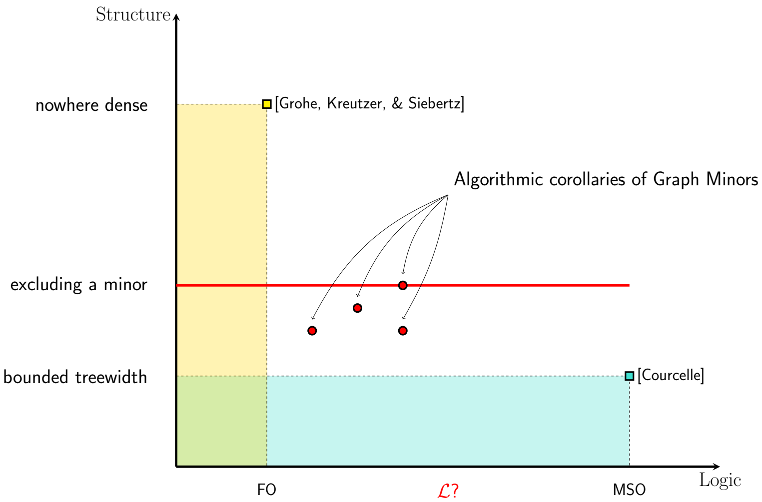

Structure
nowhere dense
[Grohe, Kreutzer, & Siebertz]
bounded treewidth
[Courcelle]
FO
$\mathcal{L}$?
MSO
Logic



Structure
nowhere dense
[Grohe, Kreutzer, & Siebertz]
Algorithmic corollaries of Graph Minors
excluding a minor
bounded treewidth
[Courcelle]
FO
$\mathcal{L}$?
MSO
Logic



- For MSO, bounded treewidth/cliquewidth is the "combinatorial limit".

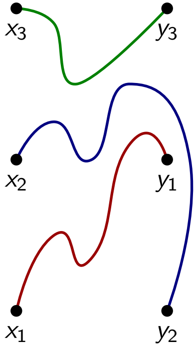


- For MSO, bounded treewidth/cliquewidth is the “combinatorial limit”.
- Logical-combinatorial *compromise* for Graph Minors?

Disjoint-paths logic (FO+dp)

$x = y \mid \textbf{adj}(x, y) \mid \text{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$

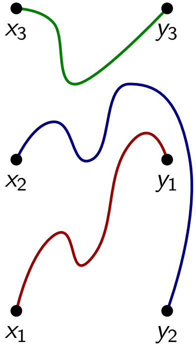
[Schirrmacher, Siebertz, & Vigny, 2021]



Disjoint-paths logic (FO+dp)

$x = y \mid \mathbf{adj}(x, y) \mid \mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$
[Schirrmacher, Siebertz, & Vigny, 2021]

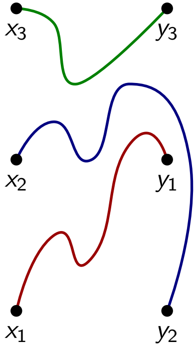
$\mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)]$ is shortcut for:
"*there exist pairwise vertex-disjoint paths connecting* x_i, y_i *for all* $i \in \{1, \ldots, k\}$"



Disjoint-paths logic (FO+dp)

$x = y \mid \textbf{adj}(x, y) \mid \text{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$
[Schirrmacher, Siebertz, & Vigny, 2021]

$\text{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)]$ is shortcut for:
"*there exist pairwise vertex-disjoint paths connecting* x_i, y_i *for all* $i \in \{1, \ldots, k\}$"

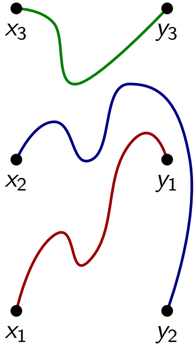


Can express: • topological minors

Disjoint-paths logic (FO+dp)

$x = y \mid \mathbf{adj}(x, y) \mid \mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$
[Schirrmacher, Siebertz, & Vigny, 2021]

$\mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)]$ is shortcut for:
"*there exist pairwise vertex-disjoint paths connecting* x_i, y_i *for all* $i \in \{1, \ldots, k\}$"

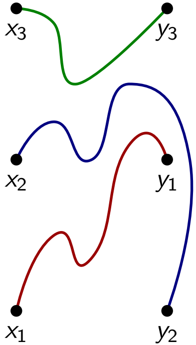


Can express:
- topological minors
- Every minor-closed property (via obstructions)

Disjoint-paths logic (FO+dp)

$x = y \mid \mathbf{adj}(x, y) \mid \mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$
[Schirrmacher, Siebertz, & Vigny, 2021]

$\mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)]$ is shortcut for:
"*there exist pairwise vertex-disjoint paths connecting* x_i, y_i *for all* $i \in \{1, \ldots, k\}$"



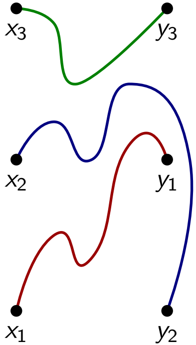
Can express:
- topological minors
- Every minor-closed property (via obstructions)

Separator logic (FO+conn) [Schirrmacher, Siebertz, & Vigny, 2021] [Bojańczyk, 2021]

Disjoint-paths logic (FO+dp)

$x = y \mid \textbf{adj}(x,y) \mid \text{dp}_k[(x_1,y_1),\ldots,(x_k,y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$
[Schirrmacher, Siebertz, & Vigny, 2021]

$\text{dp}_k[(x_1,y_1),\ldots,(x_k,y_k)]$ is shortcut for:
"there exist pairwise vertex-disjoint paths connecting x_i, y_i for all $i \in \{1,\ldots,k\}$"



Can express:
- topological minors
- Every minor-closed property (via obstructions)

Separator logic (FO+conn) [Schirrmacher, Siebertz, & Vigny, 2021] [Bojańczyk, 2021]

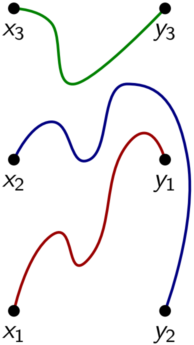
$x = y \mid \textbf{adj}(x,y) \mid \text{conn}_k(x,y,z_1,\ldots,z_k) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$

$\text{conn}_k(x,y,z_1,\ldots,z_k) = \text{dp}_{k+1}[(x,y),(z_1,z_1),\ldots,(z_k,z_k)]$

Disjoint-paths logic (FO+dp)

$x = y \mid \mathbf{adj}(x, y) \mid \mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)] \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$
[Schirrmacher, Siebertz, & Vigny, 2021]

$\mathrm{dp}_k[(x_1, y_1), \ldots, (x_k, y_k)]$ is shortcut for:
"*there exist pairwise vertex-disjoint paths connecting x_i, y_i for all $i \in \{1, \ldots, k\}$*"



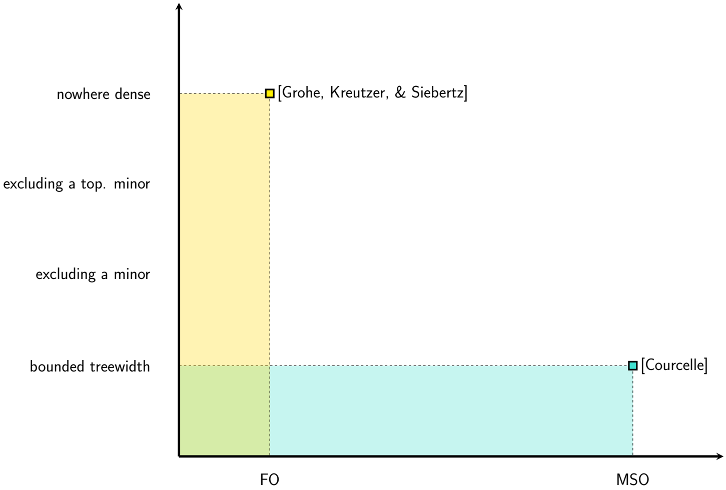
Can express:
- topological minors
- Every minor-closed property (via obstructions)

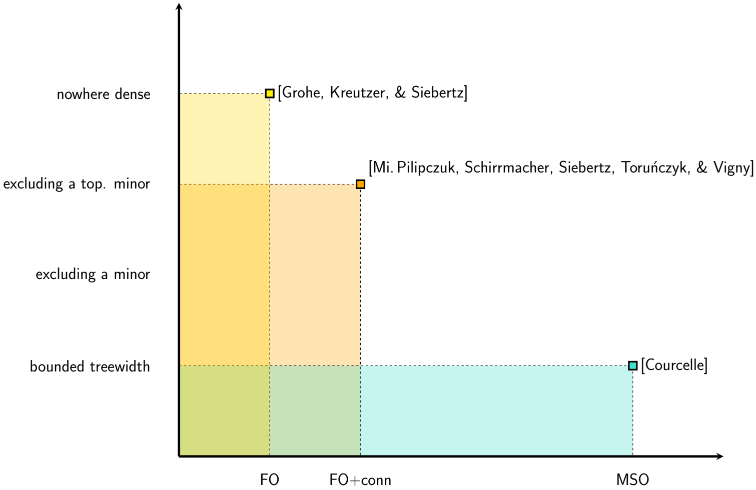
Separator logic (FO+conn) [Schirrmacher, Siebertz, & Vigny, 2021] [Bojańczyk, 2021]

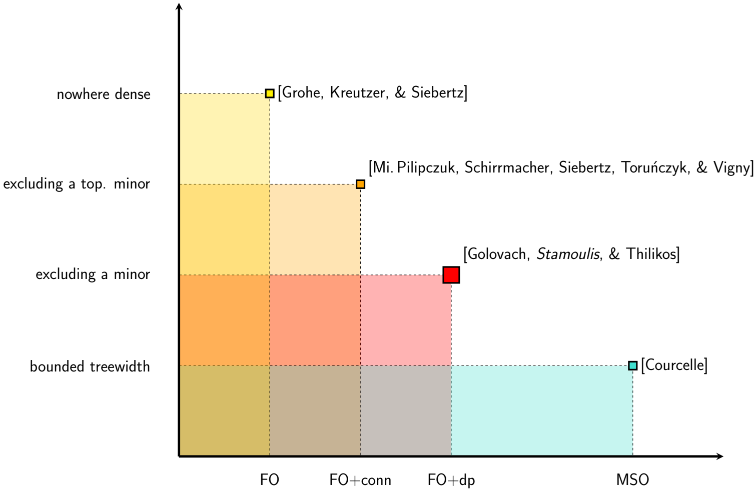
$x = y \mid \mathbf{adj}(x, y) \mid \mathrm{conn}_k(x, y, z_1, \ldots, z_k) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x \varphi \mid \forall x \varphi$

$\mathrm{conn}_k(x, y, z_1, \ldots, z_k) = \mathrm{dp}_{k+1}[(x, y), (z_1, z_1), \ldots, (z_k, z_k)]$

FO $\subseteq$ FO+conn $\subseteq$ FO+dp $\subseteq$ MSO

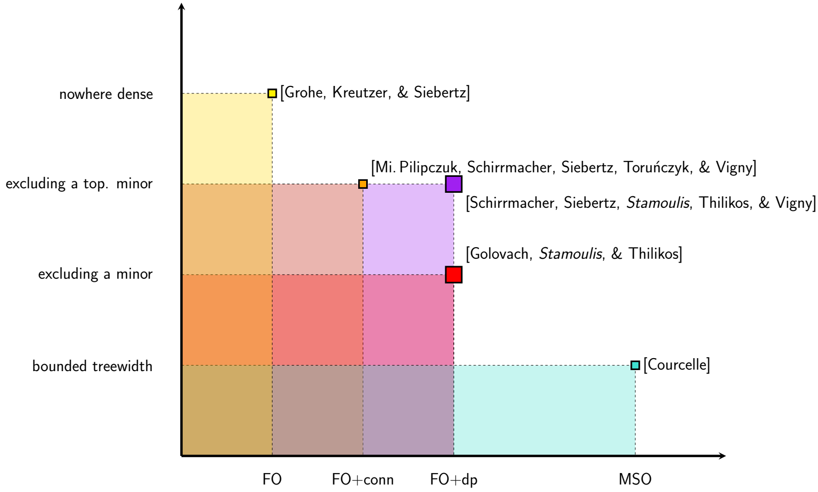

nowhere dense
[Grohe, Kreutzer, & Siebertz]
excluding a top. minor
excluding a minor
bounded treewidth
[Courcelle]
FO
MSO


nowhere dense
excluding a top. minor
excluding a minor
bounded treewidth
[Grohe, Kreutzer, & Siebertz]
[Mi. Pilipczuk, Schirrmacher, Siebertz, Toruńczyk, & Vigny]
[Courcelle]
FO
FO+conn
MSO



Model checking for FO+dp can be done in quadratic time on graphs excluding a minor.

[Golovach, *Stamoulis*, & Thilikos, 2023]



Model checking for FO+dp can be done in cubic time on graphs excluding a topological minor.

[Schirrmacher, Siebertz, *Stamoulis*, Thilikos, & Vigny, 2023+]

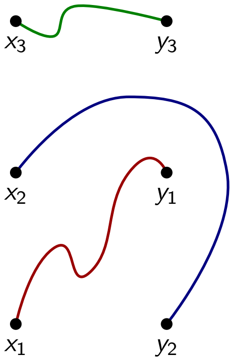
Scattered disjoint-paths logic (FO+sdp)

[Golovach, *Stamoulis*, & Thilikos, 2023]

Scattered disjoint paths predicates:

$s\text{-sdp}_k(x_1, y_1, \ldots, x_k, y_k)$

There are pairwise vertex-disjoint paths between x_i and y_i, for every $i \in \{1, \ldots, k\}$
s.t. no two vertices of two distinct paths are within distance $\leqslant s$.



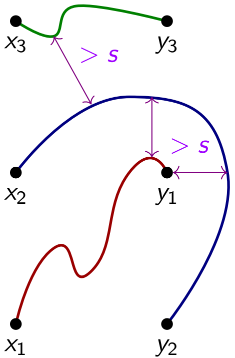
Scattered disjoint-paths logic (FO+sdp)

[Golovach, *Stamoulis*, & Thilikos, 2023]

Scattered disjoint paths predicates:

$s\text{-sdp}_k(x_1, y_1, \ldots, x_k, y_k)$

There are pairwise vertex-disjoint paths between x_i and y_i, for every $i \in \{1, \ldots, k\}$
s.t. no two vertices of two distinct paths are within distance $\leqslant s$.



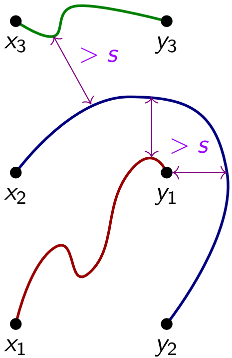
Scattered disjoint-paths logic (FO+sdp)

[Golovach, *Stamoulis*, & Thilikos, 2023]

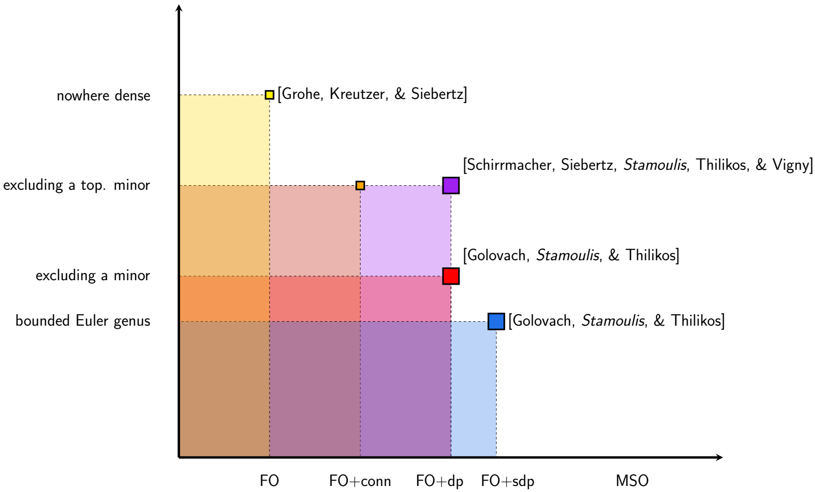
Scattered disjoint paths predicates:

$s\text{-sdp}_k(x_1, y_1, \ldots, x_k, y_k)$

There are pairwise vertex-disjoint paths between x_i and y_i, for every $i \in \{1, \ldots, k\}$
s.t. no two vertices of two distinct paths are within distance $\leqslant s$.



$\text{dp}_k(x_1, y_1, \ldots, x_k, y_k) = 0\text{-sdp}_k(x_1, y_1, \ldots, x_k, y_k)$



Model checking for FO+sdp can be done in quadratic time on graphs of bounded Euler genus.

[Golovach, *Stamoulis*, & Thilikos, 2023]

Other families of problems where irrelevant vertex technique applies?

Graph modification problems

Graph Modification Problems:

Apply a modification $\mathcal{M}$ to a graph such that the resulting graph has property $\mathcal{P}$.

Graph modification problems

Graph Modification Problems:

Apply a modification $\mathcal{M}$ to a graph such that the resulting graph has property $\mathcal{P}$.

- Typically, modification is the *deletion of a set of vertices* (modulator)

Graph modification problems

Graph Modification Problems:

Apply a modification $\mathcal{M}$ to a graph such that the resulting graph has property $\mathcal{P}$.

- Typically, modification is the *deletion of a set of vertices* (modulator)
- Modification is conditioned by some measure on the modulator:

Graph modification problems

Graph Modification Problems:

Apply a modification $\mathcal{M}$ to a graph such that the resulting graph has property $\mathcal{P}$.

- Typically, modification is the *deletion of a set of vertices* (modulator)
- Modification is conditioned by some measure on the modulator: size, structural parameter,...

Graph modification problems

Graph Modification Problems:

Apply a modification $\mathcal{M}$ to a graph such that the resulting graph has property $\mathcal{P}$.

- Typically, modification is the *deletion of a set of vertices* (modulator)
- Modification is conditioned by some measure on the modulator: size, structural parameter,...

$\hookrightarrow$ **modulator/target scheme**.

Irrelevant vertices for modulators

Graph Modification Problems: One of main research areas of Parameterized Computation.

Irrelevant vertices for modulators

Graph Modification Problems: One of main research areas of Parameterized Computation.

Irrelevant vertex technique: major role in algorithms for Graph Modification Problems

Irrelevant vertices for modulators

Graph Modification Problems: One of main research areas of Parameterized Computation.

Irrelevant vertex technique: major role in algorithms for Graph Modification Problems

Examples:

[Adler, Grohe, Kreutzer, 2008]

[Marx & Schlotter, 2012]

[Golovach, van't Hof, Paulusma, 2013]

[Kawarabayashi & Reed, 2007]

[Kawarabayashi, 2009]

[Jansen, Lokshtanov, Saurabh, 2014]

[Kociumaka & Pilipczuk, 2019]

[Fomin, Lokshtanov, Panolan, Saurabh, Zehavi, 2020]

Irrelevant vertices for modulators

Graph Modification Problems: One of main research areas of Parameterized Computation.

Irrelevant vertex technique: major role in algorithms for Graph Modification Problems

Examples:

[Adler, Grohe, Kreutzer, 2008]

[Marx & Schlotter, 2012]

[Golovach, van't Hof, Paulusma, 2013]

[Kawarabayashi & Reed, 2007]

[Kawarabayashi, 2009]

[Jansen, Lokshtanov, Saurabh, 2014]

[Kociumaka & Pilipczuk, 2019]

[Fomin, Lokshtanov, Panolan, Saurabh, Zehavi, 2020]

Challenge: Lift the application of the technique on the target to deal with the modulator.

Irrelevant vertices for modulators

Graph Modification Problems: One of main research areas of Parameterized Computation.

Irrelevant vertex technique: major role in algorithms for Graph Modification Problems

Examples:

[Adler, Grohe, Kreutzer, 2008]

[Marx & Schlotter, 2012]

[Golovach, van't Hof, Paulusma, 2013]

[Kawarabayashi & Reed, 2007]

[Kawarabayashi, 2009]

[Jansen, Lokshtanov, Saurabh, 2014]

[Kociumaka & Pilipczuk, 2019]

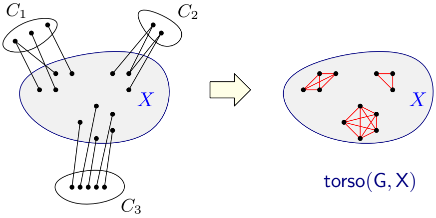
[Fomin, Lokshtanov, Panolan, Saurabh, Zehavi, 2020]

Challenge: Lift the application of the technique on the target to deal with the modulator.

But what if the modulator has *unbounded size*?

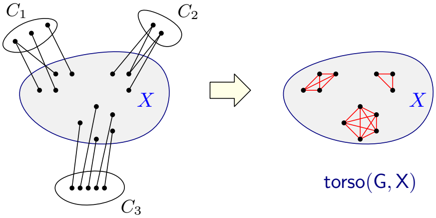
But what if the modulator has unbounded size?

Modulator: set X such that $\mathbf{p}(\text{torso}(G, X)) \leqslant k$.
Target: graph class $\mathcal{G}$.



But what if the modulator has unbounded size?

Modulator: set X such that $\mathbf{p}(\text{torso}(G, X)) \leqslant k$.
Target: graph class $\mathcal{G}$.



p=treedepth: $\mathcal{G}$-elimination distance

[Bulian & Dawar, 2017]

[Morelle, Sau, Stamoulis, Thilikos, 2023]

[Lindermayr, Siebertz, Vigny, 2020]

p=treewidth: $\mathcal{G}$-treewidth

[Eiben, Ganian, Hamm, Kwon, 2021]

[Jansen, de Kroon, Włodarczyk, 2021]

[Agrawal, Kanesh, Lokshtanov, Panolan, Ramanujan, Saurabh, Zehavi, 2022]

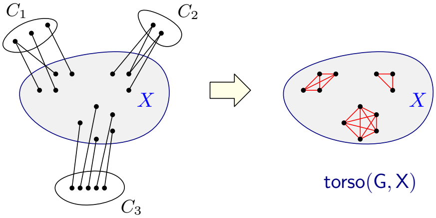
[Jansen, de Kroon, Włodarczyk, 2023]

p=bridge-depth: $\mathcal{G}$-bridge-depth

[Bougeret, Jansen, Sau, 2020]

But what if the modulator has unbounded size?

Modulator: set X such that $\mathbf{p}(\text{torso}(G, X)) \leqslant k$.
Target: graph class $\mathcal{G}$.



p=treedepth: $\mathcal{G}$-elimination distance

[Bulian & Dawar, 2017]

[Morelle, Sau, Stamoulis, Thilikos, 2023]

[Lindermayr, Siebertz, Vigny, 2020]

p=treewidth: $\mathcal{G}$-treewidth

[Eiben, Ganian, Hamm, Kwon, 2021]

[Jansen, de Kroon, Włodarczyk, 2021]

[Agrawal, Kanesh, Lokshtanov, Panolan, Ramanujan, Saurabh, Zehavi, 2022]

[Jansen, de Kroon, Włodarczyk, 2023]

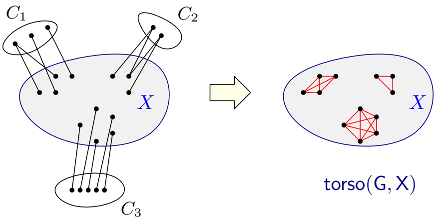
p=bridge-depth: $\mathcal{G}$-bridge-depth

[Bougeret, Jansen, Sau, 2020]

One meta-theorem that deals with all these cases?

But what if the modulator has unbounded size?

Modulator: set X such that $\mathbf{p}(\text{torso}(G, X)) \leqslant k$.
Target: graph class $\mathcal{G}$.



p=treedepth: $\mathcal{G}$-elimination distance

[Bulian & Dawar, 2017]

[Morelle, Sau, Stamoulis, Thilikos, 2023]

[Lindermayr, Siebertz, Vigny, 2020]

p=treewidth: $\mathcal{G}$-treewidth

[Eiben, Ganian, Hamm, Kwon, 2021]

[Jansen, de Kroon, Włodarczyk, 2021]

[Agrawal, Kanesh, Lokshtanov, Panolan, Ramanujan, Saurabh, Zehavi, 2022]

[Jansen, de Kroon, Włodarczyk, 2023]

p=bridge-depth: $\mathcal{G}$-bridge-depth

[Bougeret, Jansen, Sau, 2020]

One meta-theorem that deals with all these cases?

p=pathwidth, cutwidth, vertex cover, feedback vertex set, branchwidth, carving-width,...

Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Modulator
(on annotated graphs (G, X))

Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Modulator
(on annotated graphs (G, X))

Target
(on graphs)

Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Modulator (on annotated graphs (G, X)) → $\mathcal{L}_1$; Target (on graphs) → $\mathcal{L}_2$

Formulas of the form: There is a set X such that

Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

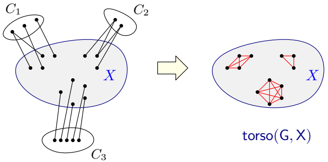
$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Modulator (on annotated graphs (G, X))

Target (on graphs)

Formulas of the form: There is a set X such that

- torso(G, X) has **bounded treewidth** and satisfies a formula $\beta \in \mathcal{L}_1$



Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

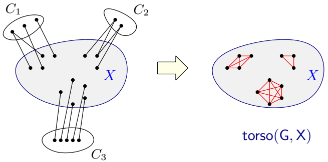
$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Modulator (on annotated graphs (G, X))

Target (on graphs)

Formulas of the form: There is a set X such that

- torso(G, X) has **bounded treewidth** and satisfies a formula $\beta \in \mathcal{L}_1$
- $G - X$ satisfies a formula $\gamma \in \mathcal{L}_2$



Compound logics

A study on unbounded size but "*structured*" modulators.

Motivation: algorithm-driven

- For logics $\mathcal{L}_1, \mathcal{L}_2$, we define

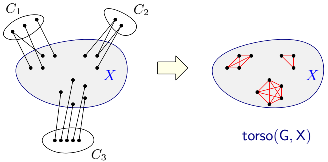
$$\mathcal{L}_1 \triangleright \mathcal{L}_2$$

Modulator
(on annotated graphs (G, X))

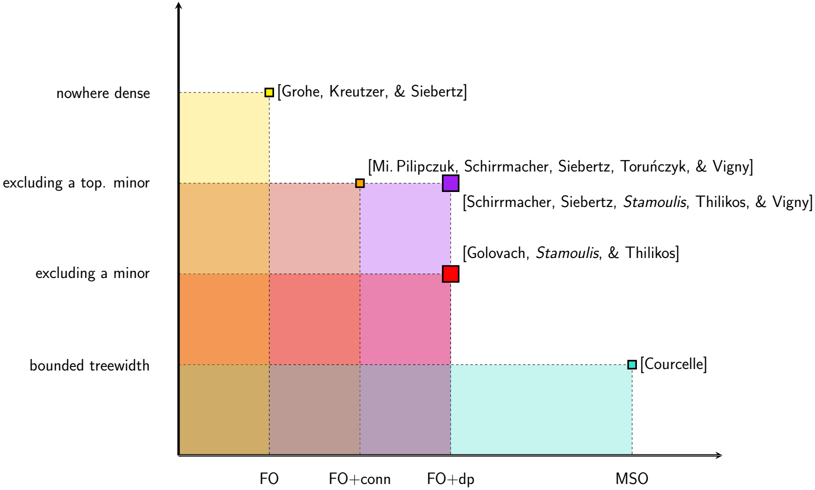
Target
(on graphs)

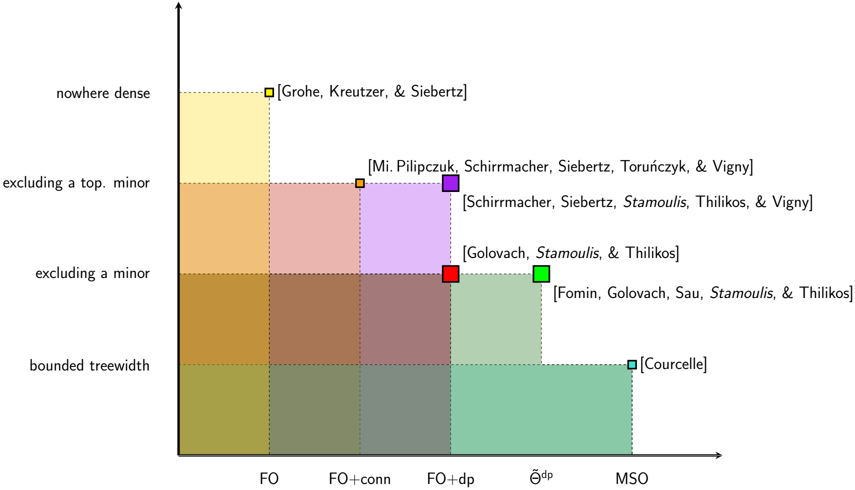
Formulas of the form: There is a set X such that

- torso(G, X) has **bounded treewidth** and satisfies a formula $\beta \in \mathcal{L}_1$
- $G - X$ satisfies a formula $\gamma \in \mathcal{L}_2$



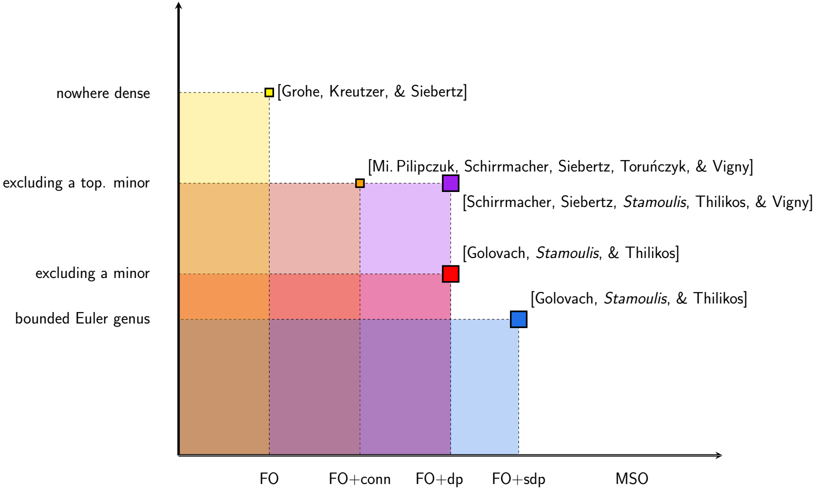
$\tilde{\Theta}^{\mathsf{dp}}$ corresponds to MSO $\triangleright$ (MSO $\triangleright$ ...(MSO $\triangleright$ FO + dp))

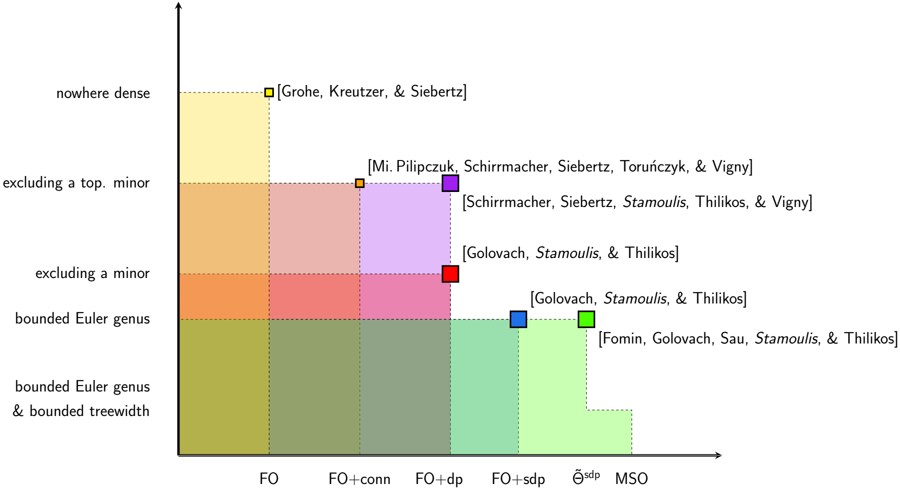

nowhere dense
excluding a top. minor
excluding a minor
bounded treewidth
FO
FO+conn
FO+dp
MSO
[Grohe, Kreutzer, & Siebertz]
[Mi. Pilipczuk, Schirrmacher, Siebertz, Toruńczyk, & Vigny]
[Schirrmacher, Siebertz, *Stamoulis*, Thilikos, & Vigny]
[Golovach, *Stamoulis*, & Thilikos]
[Courcelle]



Model checking for $\tilde{\Theta}^{dp}$ can be done in quadratic time on graphs excluding a minor.

[Fomin, Golovach, Sau, *Stamoulis*, & Thilikos, 2023]


nowhere dense
excluding a top. minor
excluding a minor
bounded Euler genus
[Grohe, Kreutzer, & Siebertz]
[Mi. Pilipczuk, Schirrmacher, Siebertz, Toruńczyk, & Vigny]
[Schirrmacher, Siebertz, *Stamoulis*, Thilikos, & Vigny]
[Golovach, *Stamoulis*, & Thilikos]
[Golovach, *Stamoulis*, & Thilikos]
FO
FO+conn
FO+dp
FO+sdp
MSO



Model checking for $\tilde{\Theta}^{\mathsf{sdp}}$ can be done in quadratic time on graphs of bounded Euler genus.

[Fomin, Golovach, Sau, *Stamoulis*, & Thilikos, 2023]

Our AMTs

- Fragments of MSO that are algorithmically well-behaved beyond bounded treewidth.

Our AMTs

- Fragments of MSO that are algorithmically well-behaved beyond bounded treewidth.

▷ (Scattered) Disjoint-paths logic *encodes paths*.

Our AMTs

- Fragments of MSO that are algorithmically well-behaved beyond bounded treewidth.

▷ (Scattered) Disjoint-paths logic *encodes paths*.

▷ Compound logics *encode modulators*.

Our AMTs

- Fragments of MSO that are algorithmically well-behaved beyond bounded treewidth.

▷ (Scattered) Disjoint-paths logic *encodes paths*.

▷ Compound logics *encode modulators*.

✓ Meta-algorithmize irrelevant vertex technique.

Our AMTs

- Fragments of MSO that are algorithmically well-behaved beyond bounded treewidth.
- ▷ (Scattered) Disjoint-paths logic *encodes paths*.
- ▷ Compound logics *encode modulators*.

✓ Meta-algorithmize irrelevant vertex technique.

[Golovach, *Stamoulis*, & Thilikos, Model-Checking for First-Order Logic with Disjoint Paths Predicates in Proper Minor-Closed Graph Classes]
SODA 2023

[Schirrmacher, Siebertz, *Stamoulis*, Thilikos, & Vigny, Model Checking Disjoint-Paths Logic on Topological-Minor-Free Graph Classes]
Unpublished

[Fomin, Golovach, Sau, *Stamoulis*, & Thilikos, Compound Logics for Modification Problems]
ICALP 2023

“Efficiency axis”

Back to Graph Minors

Goal: Identify large families of problems where running times can be improved.

Back to Graph Minors

Goal: Identify large families of problems where running times can be improved.

Modulator/target scheme:

Modulator: set of $\leqslant k$ vertices

Target: property $\mathcal{P}$

Back to Graph Minors

Goal: Identify large families of problems where running times can be improved.

Modulator/target scheme:

Modulator: set of $\leqslant k$ vertices

Target: property $\mathcal{P}$

What if $\mathcal{P}$ is characterized by exclusion of some graphs as (topological) minors?

$\mathcal{F}$-Minor-deletion **and** $\mathcal{F}$-Topological-Minor-deletion

For **finite** set of graphs $\mathcal{F}$:

$\mathcal{F}$-Minor-deletion and $\mathcal{F}$-Topological-Minor-deletion

For **finite** set of graphs $\mathcal{F}$:

$\mathcal{F}$-Minor-deletion:
Delete $\leqslant k$ vertices such that the obtained graph does not contain any $F \in \mathcal{F}$ as a minor.

$\mathcal{F}$-Minor-deletion and $\mathcal{F}$-Topological-Minor-deletion

For **finite** set of graphs $\mathcal{F}$:

$\mathcal{F}$-Minor-deletion:
Delete $\leqslant k$ vertices such that the obtained graph does not contain any $F \in \mathcal{F}$ as a minor.

$\mathcal{F}$-Topological-Minor-deletion:
Delete $\leqslant k$ vertices such that the obtained graph does not contain any $F \in \mathcal{F}$ as topological minor.

$\mathcal{F}$-Minor-deletion

[Robertson & Seymour, GM I – GM XXII]: $\mathcal{F}$-Minor-deletion is solvable (non-constructively).

$\mathcal{F}$-Minor-deletion

[Robertson & Seymour, GM I – GM XXII]: $\mathcal{F}$-Minor-deletion is solvable (non-constructively).

[Adler, Grohe, & Kreutzer, 2012]: Constructive but implicit bound on running time.

$\mathcal{F}$-Minor-deletion

[Robertson & Seymour, GM I – GM XXII]: $\mathcal{F}$-Minor-deletion is solvable (non-constructively).

[Adler, Grohe, & Kreutzer, 2012]: Constructive but implicit bound on running time.

Our results:

Bounding the obstructions

Obstructions of yes-instances of $\mathcal{F}$-Minor-deletion have size $\leqslant f(k, \mathcal{F})$

$\mathcal{F}$-Minor-deletion

[Robertson & Seymour, GM I – GM XXII]: $\mathcal{F}$-Minor-deletion is solvable (non-constructively).

[Adler, Grohe, & Kreutzer, 2012]: Constructive but implicit bound on running time.

Our results:

Bounding the obstructions

Obstructions of yes-instances of $\mathcal{F}$-Minor-deletion have size $\leqslant f(k, \mathcal{F}) = 2^{2^{2^{2^{poly_{\mathcal{F}}(k)}}}}$

$\mathcal{F}$-Minor-deletion

[Robertson & Seymour, GM I – GM XXII]: $\mathcal{F}$-Minor-deletion is solvable (non-constructively).

[Adler, Grohe, & Kreutzer, 2012]: Constructive but implicit bound on running time.

Our results:

Bounding the obstructions

Obstructions of yes-instances of $\mathcal{F}$-Minor-deletion have size $\leqslant f(k, \mathcal{F}) = 2^{2^{2^{2^{poly_{\mathcal{F}}(k)}}}}$

$\Longrightarrow$ First **explicit** upper-bound on the running time of algorithm for $\mathcal{F}$-Minor-deletion.

[Robertson & Seymour, GM I – GM XXII]: $\mathcal{F}$-Minor-deletion is solvable (non-constructively).

[Adler, Grohe, & Kreutzer, 2012]: Constructive but implicit bound on running time.

Our results:

Bounding the obstructions

Obstructions of yes-instances of $\mathcal{F}$-Minor-deletion have size $\leqslant f(k, \mathcal{F}) = 2^{2^{2^{2^{poly_{\mathcal{F}}(k)}}}}$

$\implies$ First **explicit** upper-bound on the running time of algorithm for $\mathcal{F}$-Minor-deletion.

Improved algorithm for $\mathcal{F}$-Minor-deletion

$\mathcal{F}$-Minor-deletion is solvable in time $2^{poly_{\mathcal{F}}(k)} \cdot n^2$.

$\mathcal{F}$-Topological-Minor-deletion

Not encompassed by classical Graph Minors.

$\mathcal{F}$-Topological-Minor-deletion

Not encompassed by classical Graph Minors.

[Fomin, Lokshtanov, Panolan, Saurabh, and Zehavi, 2020]: Solvable in time $\mathcal{O}_{\mathcal{F},k}(n^4)$.

$\mathcal{F}$-Topological-Minor-deletion

Not encompassed by classical Graph Minors.

[Fomin, Lokshtanov, Panolan, Saurabh, and Zehavi, 2020]: Solvable in time $\mathcal{O}_{\mathcal{F},k}(n^4)$.

Improved algorithm for $\mathcal{F}$-Topological-Minor-deletion

$\mathcal{F}$-Topological-Minor-Deletion is solvable in time $2^{\mathcal{O}_{\mathcal{F},g}(k^2)} \cdot n^2$ on graphs of Euler genus $\leqslant g$.

“Efficiency axis”

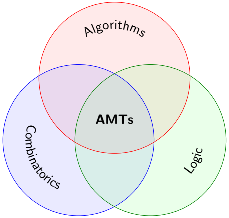
[Sau, *Stamoulis*, & Thilikos, *k*-apices of minor-closed graph classes. I. Bounding the obstructions]
Journal of Combinatorial Theory, Series B (**JCTB**), 2023

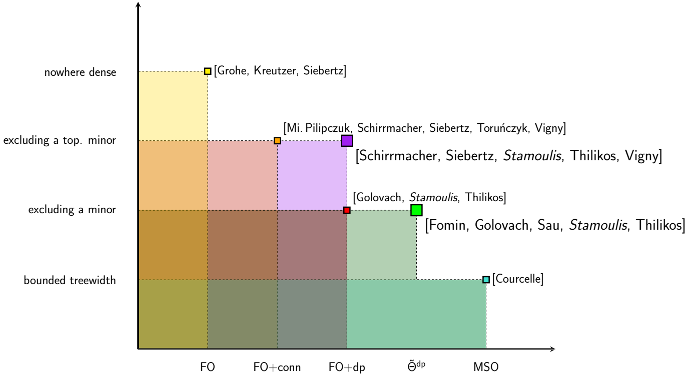
[Sau, *Stamoulis*, Thilikos, *k*-apices of minor-closed graph classes. II. Parameterized algorithms]
ICALP 2020
ACM Transactions on Algorithms (**TALG**), 2022

[Morelle, Sau, *Stamoulis*, Thilikos, Faster parameterized algorithms for modification problems to minor-closed classes]
ICALP 2023

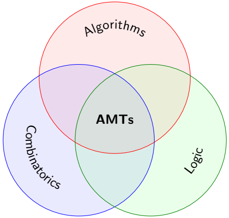
[Golovach, *Stamoulis*, Thilikos, Hitting Topological Minor Models in Planar Graphs is Fixed Parameter Tractable]
SODA 2020
ACM Transactions on Algorithms (**TALG**), 2023

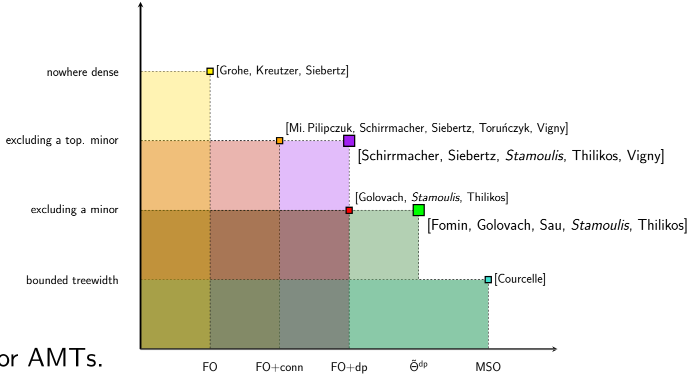
Recap of results of the thesis





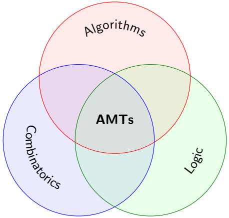
Recap of results of the thesis

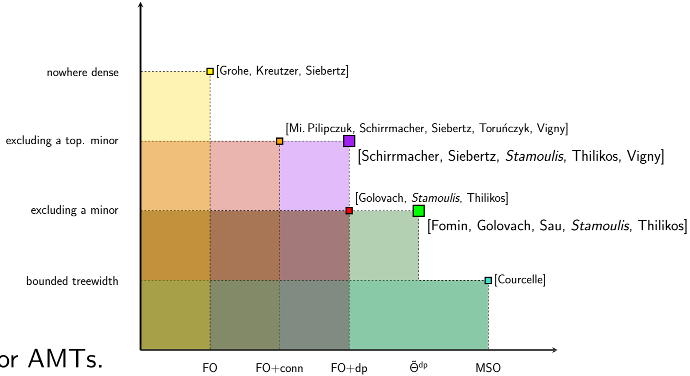




- Combinatorial & algorithmic support for AMTs.

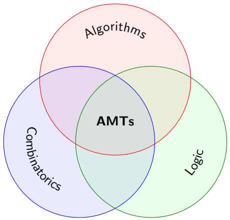
Recap of results of the thesis

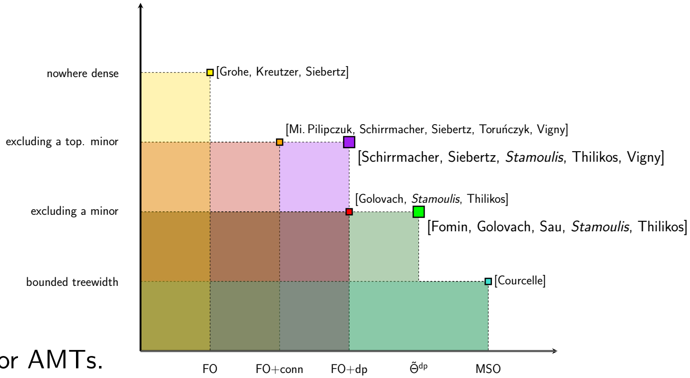




- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique.

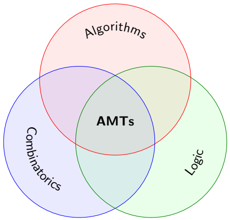
Recap of results of the thesis

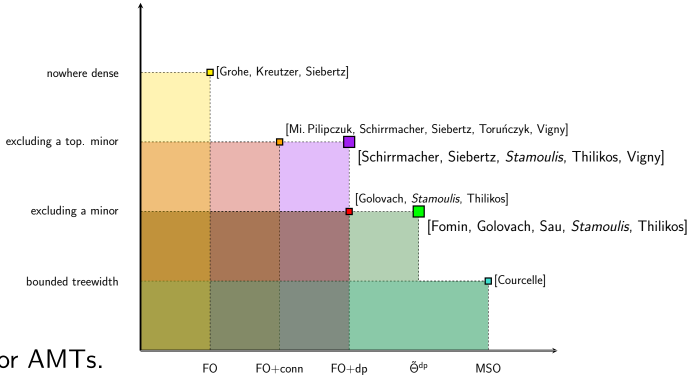




- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique.
- Advance in the efficiency dimension of AMTs.

Recap of results of the thesis

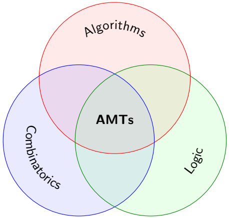


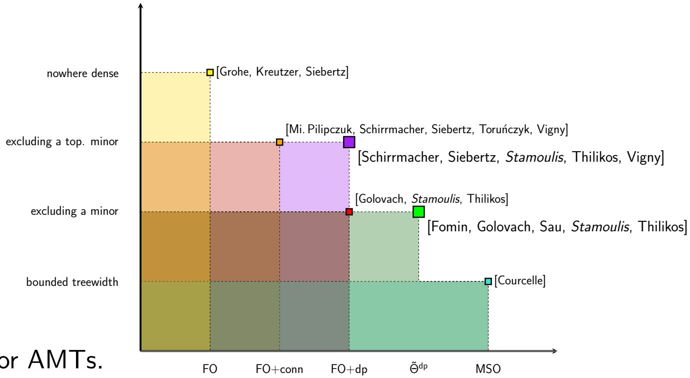


- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique.
- Advance in the efficiency dimension of AMTs.

What was needed:

Recap of results of the thesis



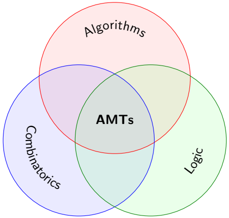


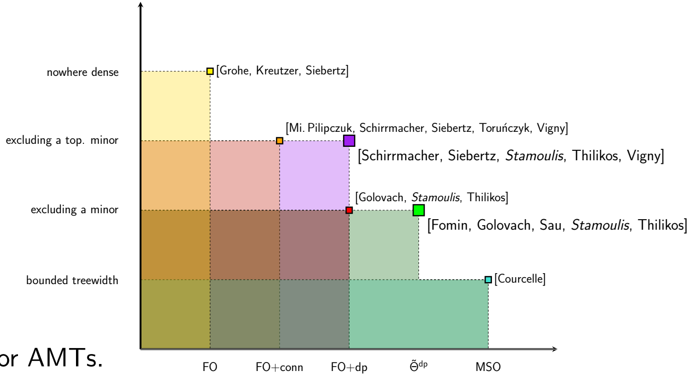
- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique.
- Advance in the efficiency dimension of AMTs.

What was needed:

▷ New combinatorial tools

Recap of results of the thesis





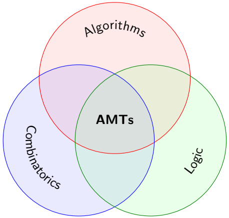
- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique.
- Advance in the efficiency dimension of AMTs.

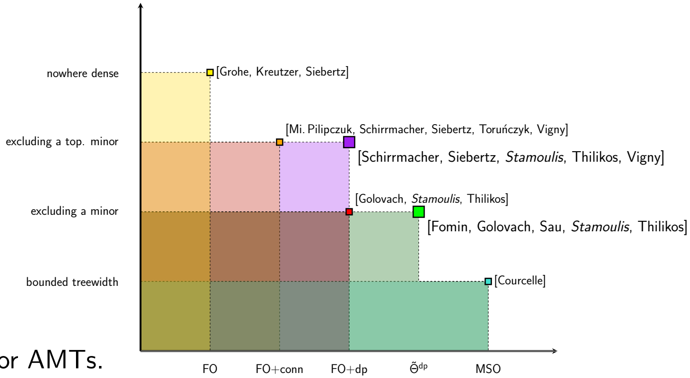
What was needed:

▷ New combinatorial tools

▷ Understanding common logical description of problems (*algorithmic paradigm of* Simplification)

Recap of results of the thesis





- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique.
- Advance in the efficiency dimension of AMTs.

What was needed:

▷ New combinatorial tools
▷ Understanding common logical description of problems (*algorithmic paradigm of* Simplification)
▷ New ideas to obtain efficient algorithms.

Outline of some ingredients of our proofs

How to meta-algorithmize Simplification?

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

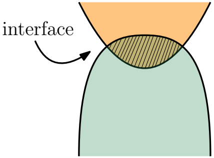
Subroutine: Recursively compute the MSO-type of the instance.

(all satisfiable MSO-formulas of certain # of quantifiers)

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.

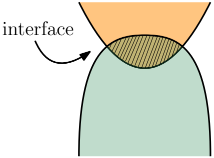


- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



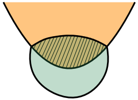
- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

How to meta-algorithmize Simplification**?**

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



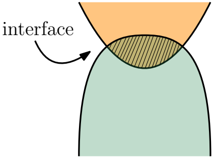
- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

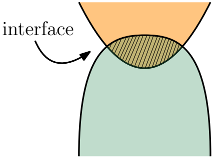
▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

How to meta-algorithmize Simplification**?**

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

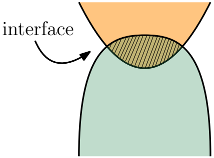
Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

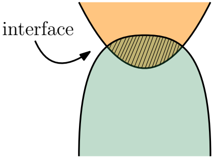
For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

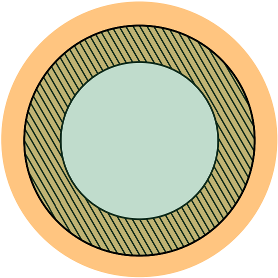
▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

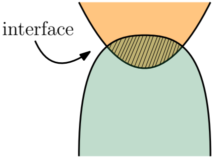
- Compositionality on unbounded size interface (in flat part).



How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

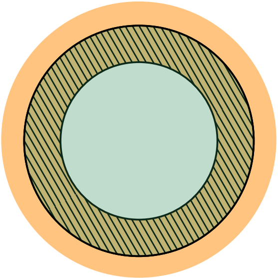
▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

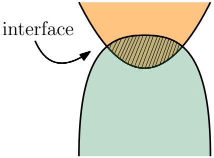
- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.



How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

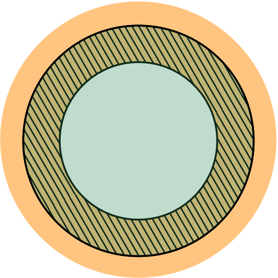
Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.

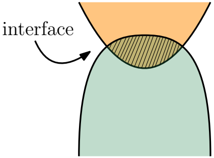
Local simplifications are global simplifications.



How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

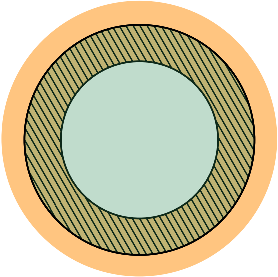
▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.



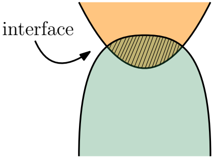
Local simplifications are global simplifications.

▷ Simplify locally using Courcelle's theorem.

How to meta-algorithmize Simplification**?**

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

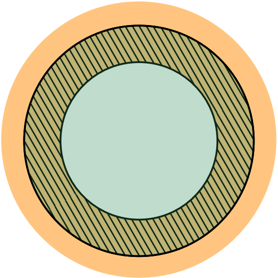
For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.

Local simplifications are global simplifications.

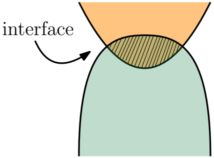
▷ Simplify locally using Courcelle's theorem.



How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

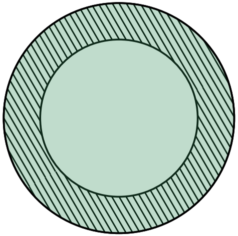
▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

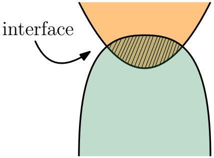
- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.



Local simplifications are global simplifications.

▷ Simplify locally using Courcelle's theorem.

How to meta-algorithmize Simplification?



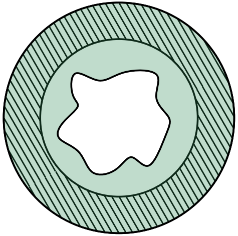
What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.

- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.



For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.

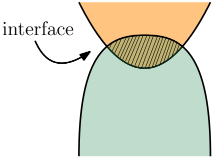
Local simplifications are global simplifications.

▷ Simplify locally using Courcelle's theorem.

How to meta-algorithmize Simplification?

What is Courcelle's theorem?

Subroutine: Recursively compute the MSO-type of the instance.



- Compositionality of MSO on small size interface. [Feferman-Vaught theorem]

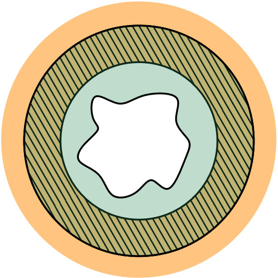
▷ *Simplification:* Reduction to type-representative of small size.

Recursion & Compositionality & Simplification = Dyn. Prog.

For our AMTs: *"Local-to-Global" approach*

Idea: Simplify flat parts of the input.

- Compositionality on unbounded size interface (in flat part).
  ↪ *Combing Lemma*.

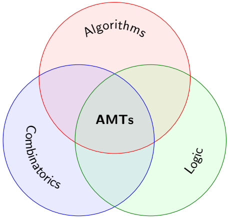


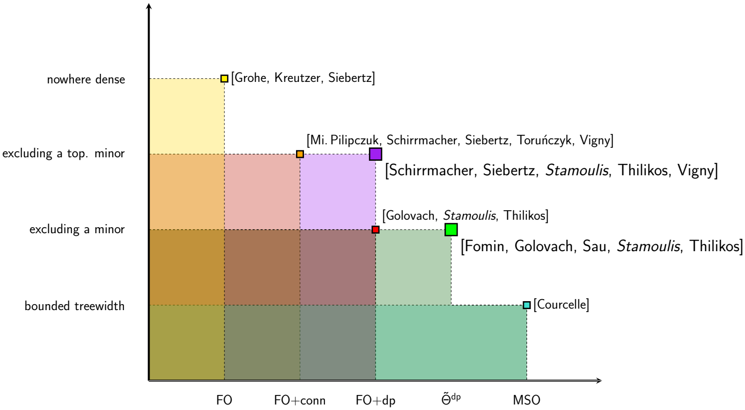
Local simplifications are global simplifications.

▷ Simplify locally using Courcelle's theorem.

Conclusions & Perspectives

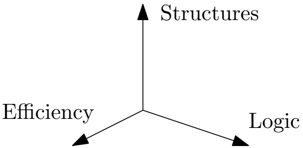
Conclusion





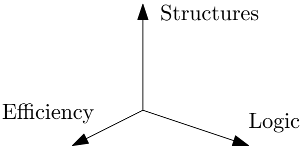
- Combinatorial & algorithmic support for AMTs.
- AMTs abstractizing irrelevant vertex technique (*algorithmic paradigm of* Simplification).
- Advance in the efficiency dimension of AMTs.

What to do next?



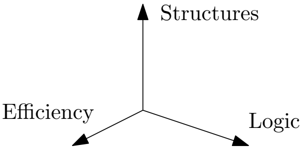
What to do next?

- Can our AMTs be generalized to more general classes?



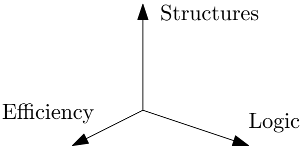
What to do next?

- Can our AMTs be generalized to more general classes?
- What is the "logical" limit on minor-closed classes?

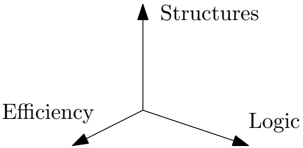


What to do next?

- ▷ Can our AMTs be generalized to more general classes?
- ▷ What is the “logical” limit on minor-closed classes?
- ▷ **Two challenges in the “efficiency dimension”**:

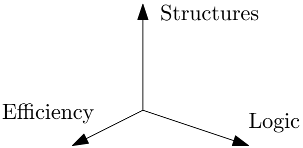


What to do next?



- ▷ Can our AMTs be generalized to more general classes?
- ▷ What is the "logical" limit on minor-closed classes?
- ▷ **Two challenges in the "efficiency dimension"**:
  - Break the barrier of $\mathcal{O}(n^2)$-time for irrelevant vertex technique?

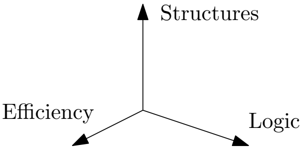
What to do next?



- ▷ Can our AMTs be generalized to more general classes?
- ▷ What is the "logical" limit on minor-closed classes?
- ▷ **Two challenges in the "efficiency dimension"**:
  - Break the barrier of $\mathcal{O}(n^2)$-time for irrelevant vertex technique?

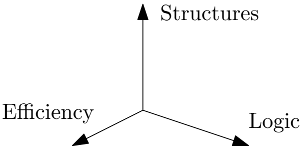
$$\mathcal{O}(n^{2-\varepsilon})? \quad \mathcal{O}(n^{1+\varepsilon})? \quad \mathcal{O}(n^{1+o(1)})? \quad \mathcal{O}(n \cdot \mathrm{polylog}(n))? \quad \mathcal{O}(n)?$$

What to do next?



- ▷ Can our AMTs be generalized to more general classes?
- ▷ What is the "logical" limit on minor-closed classes?
- ▷ **Two challenges in the "efficiency dimension"**:
  - Break the barrier of $\mathcal{O}(n^2)$-time for irrelevant vertex technique?
  - *Elementary* model checking? Running-time with elementary dependency on $|\varphi|$.

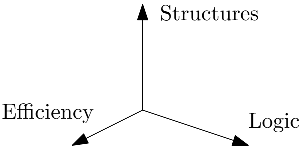
What to do next?



- ▷ Can our AMTs be generalized to more general classes?
- ▷ What is the "logical" limit on minor-closed classes?
- ▷ **Two challenges in the "efficiency dimension"**:
  - Break the barrier of $\mathcal{O}(n^2)$-time for irrelevant vertex technique?
  - *Elementary* model checking? Running-time with elementary dependency on $|\varphi|$.

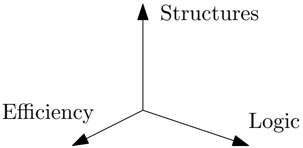
$$\left.2^{2^{\cdot^{\cdot^{2^{|\varphi|}}}}}\right\}\text{ depends on } |\varphi| \quad \to \quad \left.2^{2^{\cdot^{\cdot^{2^{|\varphi|}}}}}\right\}\textbf{ does not}\text{ depend on } |\varphi|$$

What to do next?



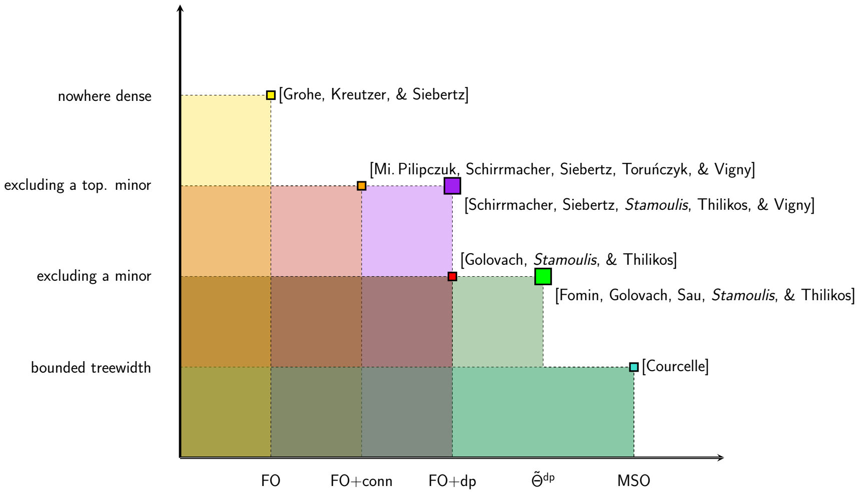
- ▷ Can our AMTs be generalized to more general classes? WiP
- ▷ What is the "logical" limit on minor-closed classes? WiP
- ▷ **Two challenges in the "efficiency dimension"**:
  - Break the barrier of $\mathcal{O}(n^2)$-time for irrelevant vertex technique? WiP
  - *Elementary* model checking? Running-time with elementary dependency on $|\varphi|$. WiP

What **we do** next?



- ▷ Can our AMTs be generalized to more general classes?
- ▷ What is the "logical" limit on minor-closed classes?
- ▷ **Two challenges in the "efficiency dimension"**:
  - Break the barrier of $\mathcal{O}(n^2)$-time for irrelevant vertex technique?
  - *Elementary* model checking? Running-time with elementary dependency on $|\varphi|$.

AMTs in Distributed Computing? Dynamic algorithms? Query enumeration?


nowhere dense
excluding a top. minor
excluding a minor
bounded treewidth
[Grohe, Kreutzer, & Siebertz]
[Mi. Pilipczuk, Schirrmacher, Siebertz, Toruńczyk, & Vigny]
[Schirrmacher, Siebertz, *Stamoulis*, Thilikos, & Vigny]
[Golovach, *Stamoulis*, & Thilikos]
[Fomin, Golovach, Sau, *Stamoulis*, & Thilikos]
[Courcelle]
FO
FO+conn
FO+dp
$\tilde{\Theta}^{\mathsf{dp}}$
MSO

Thank you!

Other research projects during Ph.D. studies (not included in the thesis)

[Fomin, Golovach, Korhonen, Simonov, *Stamoulis*. Fixed-Parameter Tractability of Maximum Colored Path and Beyond]

SODA 2023

[Fomin, Golovach, Korhonen, Lokshtanov, *Stamoulis*. Shortest Cycles With Monotone Submodular Costs]

SODA 2023

ACM Transactions on Algorithms (**TALG**), 2023

[Fomin, Golovach, Korhonen, *Stamoulis*. Computing paths of large rank in planar frameworks deterministically]

ISAAC 2023

[Diner, Giannopoulou, *Stamoulis*, Thilikos. Block Elimination Distance]

WG 2021

Graphs and Combinatorics (**GCOM**), 2022

[Kontogeorgiou, Leivaditis, Psaromiligkos, *Stamoulis*, Zoros. Branchwidth is (1,g)-self-dual]

Under revision