

# Algorithmic aspects of graph classes with certificates using patterns

Michel Habib (IRIF, Paris), Laurent Feuilloley (IRIS, Lyon)

Tempogral, Poitiers, November 24, 2022



# Schedule of the talk

Introduction

Patterns on 3 nodes

Patterns on 4 nodes

Geometric classes

Finding an ordering versus certifying it

Some Algorithmic Applications

Conclusions and perspectives

## Introduction

Patterns on 3 nodes

Patterns on 4 nodes

Geometric classes

Finding an ordering versus certifying it

Some Algorithmic Applications

Conclusions and perspectives

## Our initial question on patterns

- ▶ Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?

## Our initial question on patterns

- ▶ Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?
- ▶ Laurent did a summer internship in Santiago with R. Correa and obtain some results published in 2015.

## Our initial question on patterns

- ▶ Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?
- ▶ Laurent did a summer internship in Santiago with R. Correa and obtain some results published in 2015.
- ▶ Then during his PhD directed by Pierre Fraigniaud we obtain our first results and he succeeded to convince Pierre to consider patterns in distributed environment for local certification.

## Our initial question on patterns

- ▶ Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?
- ▶ Laurent did a summer internship in Santiago with R. Correa and obtain some results published in 2015.
- ▶ Then during his PhD directed by Pierre Fraigniaud we obtain our first results and he succeeded to convince Pierre to consider patterns in distributed environment for local certification.
- ▶ Now it is an hot subject (lot of new results) and we hope that it will become a common tool to study graph classes.

## Examples to start with

Chordal graphs: every cycle of length  $\geq 4$  has a chord.

Theorem Dirac 1961

A graph is chordal iff it admits a simplicial elimination scheme.



## Examples to start with

Chordal graphs: every cycle of length  $\geq 4$  has a chord.

Theorem Dirac 1961

A graph is chordal iff it admits a simplicial elimination scheme.

Simplicial elimination scheme

A vertex is simplicial if its neighbourhood is a clique.

$\sigma = [x_1 \dots x_i \dots x_n]$  is a simplicial elimination scheme if  $x_i$  is simplicial in the subgraph  $G_i = G[\{x_i \dots x_n\}]$

## Examples to start with

Chordal graphs: every cycle of length  $\geq 4$  has a chord.

Theorem Dirac 1961

A graph is chordal iff it admits a simplicial elimination scheme.

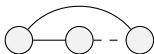
Simplicial elimination scheme

A vertex is simplicial if its neighbourhood is a clique.

$\sigma = [x_1 \dots x_i \dots x_n]$  is a simplicial elimination scheme if  $x_i$  is simplicial in the subgraph  $G_i = G[\{x_i \dots x_n\}]$

In other words:

A graph is chordal iff it admits vertex ordering without



# Comparability graphs

## Comparability graph

A graph  $G = (V, E)$  is a comparability graph if and only if  $G$  can be transitively oriented.

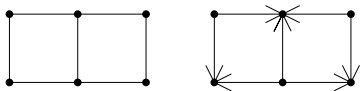


Figure: A comparability graph  $G$  and a transitive orientation of  $G$ .

# Comparability graphs

## Comparability graph

A graph  $G = (V, E)$  is a comparability graph if and only if  $G$  can be transitively oriented.

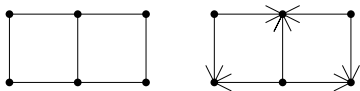


Figure: A comparability graph  $G$  and a transitive orientation of  $G$ .

## Cocomparability graph

A graph  $G = (V, E)$  is a cococomparability graph if and only if  $\overline{G}$  is a comparability graph.

# Cocomparability graphs

Definition:

For a total ordering  $\tau$  of the set of vertices, an umbrella is a triple of vertices  $a, b, c \in X$  such that:  $a <_{\tau} b <_{\tau} c$  and  $ac \in E$  and  $ab, bc \notin E$ .

A co-comparability (co-comp for short) ordering is an umbrella-free total ordering of the vertices of  $G$ .



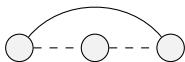
$a, b, c$ , an umbrella

# Cocomparability graphs

Definition:

For a total ordering  $\tau$  of the set of vertices, an umbrella is a triple of vertices  $a, b, c \in X$  such that:  $a <_{\tau} b <_{\tau} c$  and  $ac \in E$  and  $ab, bc \notin E$ .

A co-comparability (co-comp for short) ordering is an umbrella-free total ordering of the vertices of  $G$ .



$a, b, c$ , an umbrella

Remark:

A cocomp ordering corresponds to a linear extension of a transitive orientation of the complement.

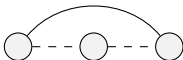
## Vertex orderings

Vertex orderings are very useful for recognition algorithms, when a graph class can be defined by the existence of an ordering avoiding some patterns.

## Vertex orderings

Vertex orderings are very useful for recognition algorithms, when a graph class can be defined by the existence of an ordering avoiding some patterns.

- ▶ A graph is a co-comparability graph iff it admits a cocomp ordering.

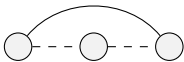




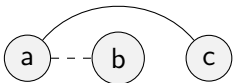
## Vertex orderings

Vertex orderings are very useful for recognition algorithms, when a graph class can be defined by the existence of an ordering avoiding some patterns.

- ▶ A graph is a co-comparability graph iff it admits a cocomp ordering.



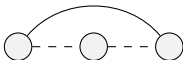
- ▶ A graph is an interval graph iff it admits an interval ordering



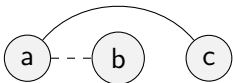
## Vertex orderings

Vertex orderings are very useful for recognition algorithms, when a graph class can be defined by the existence of an ordering avoiding some patterns.

- ▶ A graph is a co-comparability graph iff it admits a cocomp ordering.



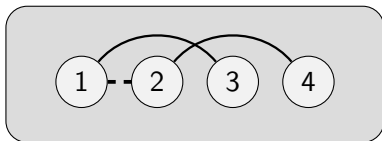
- ▶ A graph is an interval graph iff it admits an interval ordering



- ▶ A graph is a proper interval graph iff it admits a proper interval ordering



## Formalization of our model of ordered patterns



For an ordered subgraph to match the pattern:

- ▶ plain edges must be present,
- ▶ dashed edges must be absent,
- ▶ non-edges have no constraint (i.e., both are forbidden).

## Vertex ordering characterizations

A graph is a 



there exists a vertex ordering  
that avoids:

Pattern 1

Pattern 2

Pattern 3

...

First introduced by [Damaschke 90]

- ▶ The patterns we consider here are supposed to be finite.

- ▶ The patterns we consider here are supposed to be finite.
- ▶ We can only define hereditary classes of graphs using this definition.

# Examples

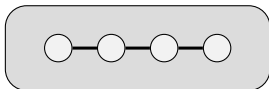
A zoo of classes

## $k$ -colourable

A graph is a 3-colourable








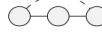
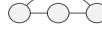



there exists a vertex ordering  
that avoids:





# From Damaschke 90

Graph Class	Forbidden Pattern
Linear forests	
Stars	
Interval graphs	
Split graphs	
Forest	
Bipartite graphs	
Chordal graphs	
Comparability graphs	
Triangle-free graphs	
At most two nodes	

# We came interested to this topic via efficient algorithms

- ▶ Most efficient recognition algorithms provide such an ordering.

## We came interested to this topic via efficient algorithms

- ▶ Most efficient recognition algorithms provide such an ordering.
- ▶ True for chordal, proper interval, interval, cocomparability . . .

## We came interested to this topic via efficient algorithms

- ▶ Most efficient recognition algorithms provide such an ordering.
- ▶ True for chordal, proper interval, interval, cocomparability . . .
- ▶ Often using a series of graph searches such as BFS, LexBFS, LexDFS . . . (whose ordering of the vertices can also be characterized using patterns).

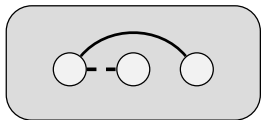
## **Some basic facts on patterns**

# Complement

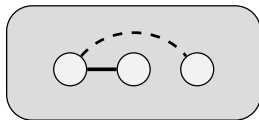
Interval

$\rightleftharpoons$

Co-interval



$\rightleftharpoons$



Inversion of  
dashed/plain  
edges

$\Leftrightarrow$

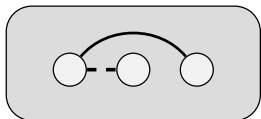
Complement  
class

# Inclusion

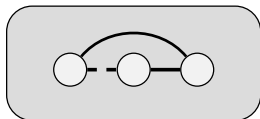
Interval

$\subseteq$

Chordal



$\supseteq$



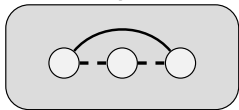
Inclusion  
of patterns

$\Rightarrow$

Inclusion  
of classes

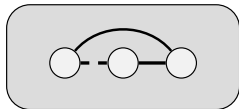
# Pattern splitting

Co-comparability



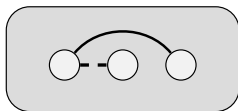
&

Chordal



$\subseteq$

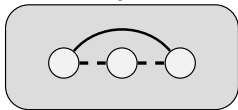
Interval





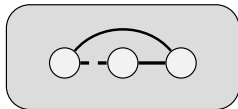
# Pattern splitting

Co-comparability



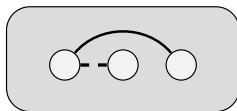
&

Chordal

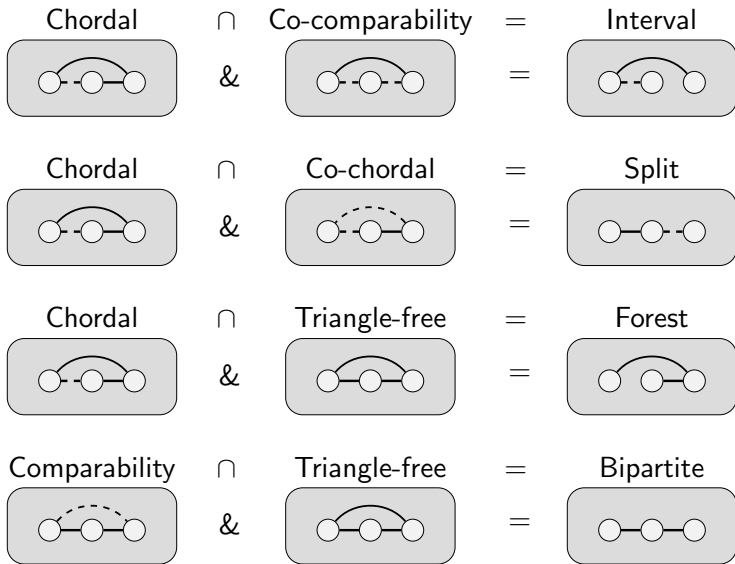


=

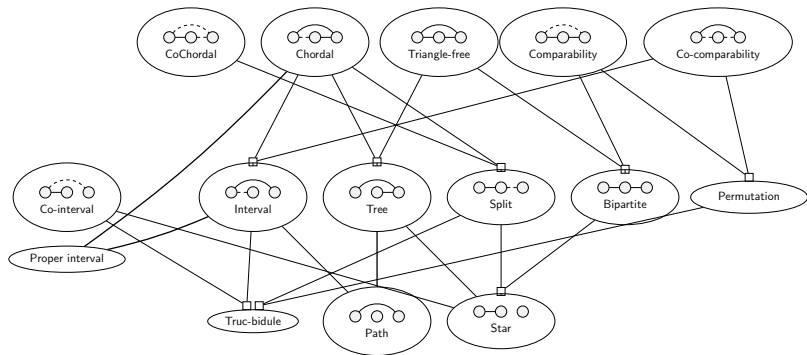
Interval



# Pattern splitting



# Diagram



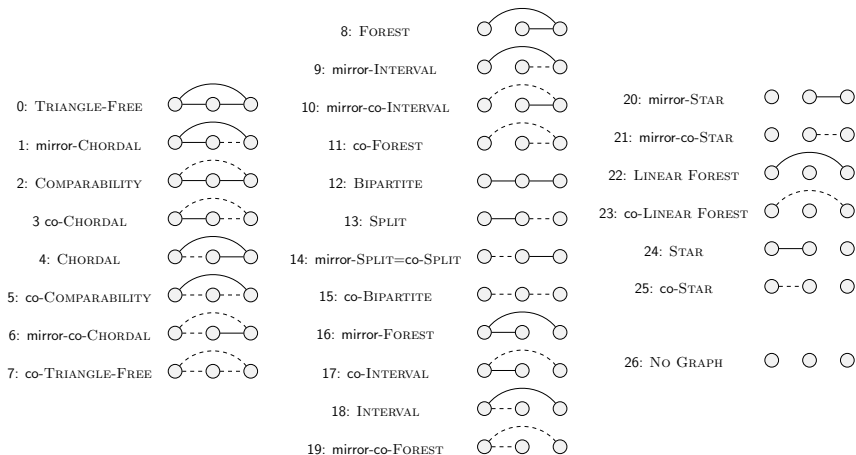


Figure: The 27 patterns on three nodes. By convention since mirror-SPLIT=co-SPLIT, we will ignore the pattern mirror-SPLIT.

# Recognition

# NP

Recognition of classes defined by forbidden patterns  
is in NP.

The ordering can be checked in polytime.

Introduction

**Patterns on 3 nodes**

Patterns on 4 nodes

Geometric classes

Finding an ordering versus certifying it

Some Algorithmic Applications

Conclusions and perspectives

## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step



## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step
  - ▶ OK with one pattern (easy: already known Damaschke 90)

## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step
  - ▶ OK with one pattern (easy: already known Damaschke 90)
  - ▶ OK with two patterns (still easy)

## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step
  - ▶ OK with one pattern (easy: already known Damaschke 90)
  - ▶ OK with two patterns (still easy)
  - ▶ OK with 3 patterns ....(tedious even if some classes are very simple)

## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step
  - ▶ OK with one pattern (easy: already known Damaschke 90)
  - ▶ OK with two patterns (still easy)
  - ▶ OK with 3 patterns ... (tedious even if some classes are very simple)
- ▶ a general algorithm [Hell, Mohar, Rafiey, 2014] (quite hard to understand with a lot of cases) using a 2-SAT based algorithm in  $O(mn)$ ;

## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step
  - ▶ OK with one pattern (easy: already known Damaschke 90)
  - ▶ OK with two patterns (still easy)
  - ▶ OK with 3 patterns ....(tedious even if some classes are very simple)
- ▶ a general algorithm [Hell, Mohar, Rafiey, 2014] (quite hard to understand with a lot of cases) using a 2-SAT based algorithm in  $O(mn)$ ;
- ▶ It is worth to realize that Golumbic's comparability graphs recognition algorithm can be translated in a 2-SAT fashion.

## On three nodes

### **Theorem Hell, Mohar, Rafiey 2014:**

Classes defined by any set of patterns on three nodes can be recognized in polynomial-time.

Proof history:

- ▶ step by step
  - ▶ OK with one pattern (easy: already known Damaschke 90)
  - ▶ OK with two patterns (still easy)
  - ▶ OK with 3 patterns ....(tedious even if some classes are very simple)
- ▶ a general algorithm [Hell, Mohar, Rafiey, 2014] (quite hard to understand with a lot of cases) using a 2-SAT based algorithm in  $O(mn)$ ;
- ▶ It is worth to realize that Golumbic's comparability graphs recognition algorithm can be translated in a 2-SAT fashion.
- ▶ Could we find a general algorithm with a simpler proof?

Theorem (L. Feuilloley, MH 2018)

*Up to complement and basic operations, the non-trivial classes that can be characterized by a set of patterns on three vertices are the following.*

- |                   |                       |                                     |
|-------------------|-----------------------|-------------------------------------|
| 1. forests        | 10. permutation       | 18. augmented clique                |
| 2. linear forests | 11. threshold         |                                     |
| 3. stars          | 12. proper interval   | 19. bipartite permutation           |
| 4. interval       | 13. caterpillar       | 20. triangle-free $\cap$ co-chordal |
| 5. split          | 14. trivially perfect |                                     |
| 6. bipartite      | 15. bipartite chain   | 21. clique                          |
| 7. chordal        | 16. 2-star            | 22. complete bipartite              |
| 8. comparability  | 17. 1-split           |                                     |
| 9. triangle-free  |                       |                                     |

## A surprising result

Only 22 well-known graph classes !

Cographs or Distance Hereditary are not in the list.

In fact the proof is a tedious case by case study of the 87 split-minimal families of patterns on 3 vertices (out of the  $2^{2^7}$  possible cases via a proved program).



# First Consequences

## Corollary

*All classes of graphs defined with sets of patterns on 3 vertices can be recognized in  $O(n^{2,3727})$ .*

## Linear

All recognition are linear-time using graph searches, except for Triangle-free and Comparability (and their complement). **Checking if an ordering avoid the pattern is the bottleneck part for these 4 cases.**

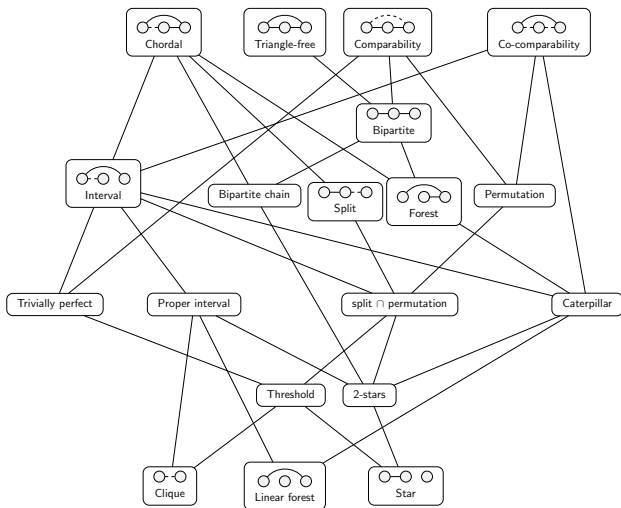


Figure: Partial inclusion diagram of the classes that appear in Theorem 1. More refined diagrams can be found in the next diagrams.

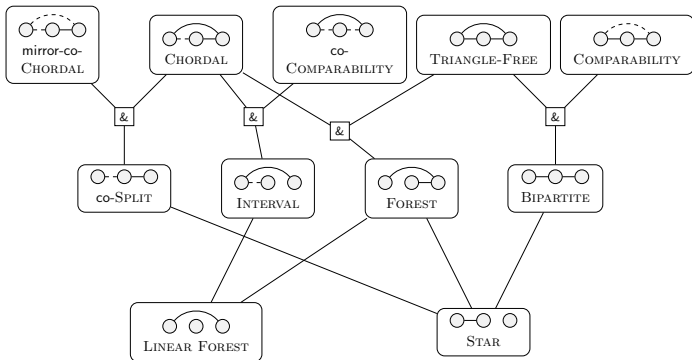


Figure: Refinement of Figure 3 in which we represent the cases where  $P = P_1 \& P_2$  and the union-intersection property holds by a label  $\&$  link to  $P_1$  and  $P_2$  above, and  $P$  below.

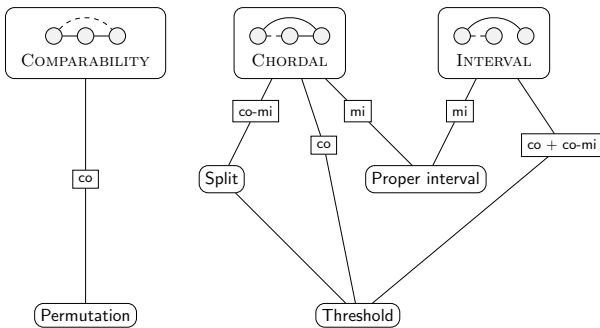


Figure: The edges labeled with “co” mean: the family made by taking the pattern on the top endpoint and its complement characterize the class below. The edges labeled with “mi” mean the same but with mirror instead of complement. And “co-mi” designate both operations. The + means that the edge has both labels.

Introduction

Patterns on 3 nodes

**Patterns on 4 nodes**

Geometric classes

Finding an ordering versus certifying it

Some Algorithmic Applications

Conclusions and perspectives

## Patterns on 4 nodes

Bigger patterns are needed to characterize other well structured classes of graphs, as for example cographs.

## The particular case of cographs

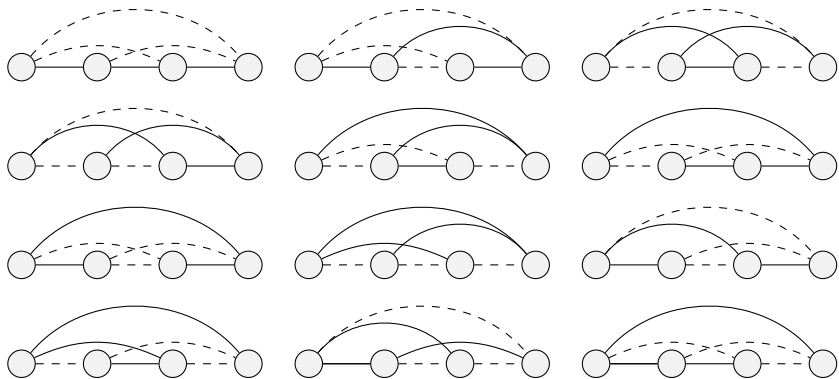


Figure: The 12 different orderings of a  $P_4$ .

Any hereditary class with a finite number of forbidden subgraphs, can also be defined using a finite number of patterns.

But three patterns are enough for cographs.

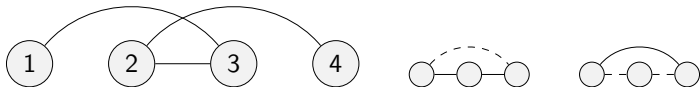


Figure: Forbidden patterns of cographs.



# Twins

**Twins**  $x, y \in V$  are false- (resp. true-) **twins** if  
 $N(x) = N(y)$  (resp.  $N(x) \cup \{x\} = N(y) \cup \{y\}$ ).  
 $x, y$  are false twins in  $G$  iff  $x, y$  are true twins in  $\overline{G}$ .

# Twins

**Twins**  $x, y \in V$  are false- (resp. true-) **twins** if

$$N(x) = N(y) \text{ (resp. } N(x) \cup \{x\} = N(y) \cup \{y\}\text{)}.$$

$x, y$  are false twins in  $G$  iff  $x, y$  are true twins in  $\overline{G}$ .

**Elimination scheme**  $G$  is a cograph iff there exists an ordering of the vertices

s.t.  $x_i$  has a twin (false or true) in  $G\{x_{i+1}, \dots, x_n\}$

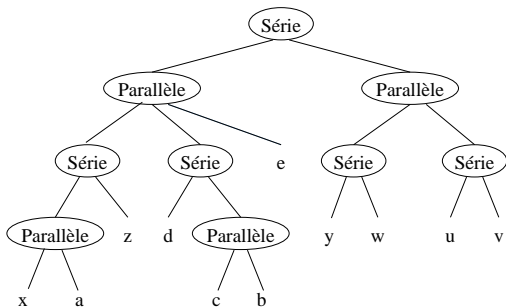
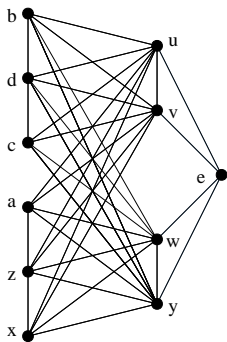
# Twins

**Twins**  $x, y \in V$  are false- (resp. true-) **twins** if  
 $N(x) = N(y)$  (resp.  $N(x) \cup \{x\} = N(y) \cup \{y\}$ ).  
 $x, y$  are false twins in  $G$  iff  $x, y$  are true twins in  $\overline{G}$ .

**Elimination scheme**  $G$  is a cograph iff there exists an ordering of the vertices  
s.t.  $x_i$  has a twin (false or true) in  $G\{x_{i+1}, \dots, x_n\}$

**Fact** Such an elimination ordering avoids the 3 forbidden patterns of cographs.

# An example of cograph



$\tau = x, a, cb, z, d, y, u, v, w, e$   
is such an ordering.

## Remarks

- ▶ But such an ordering cannot always be obtained using a standard graph search (refinement of the generic search).

## Remarks

- ▶ But such an ordering cannot always be obtained using a standard graph search (refinement of the generic search).
- ▶ For cographs to check is an ordering is an elimination sequence requires linear-time, either using the elimination ordering or the fact that cographs are permutation graphs (the 2 patterns on three nodes) without P4 patterns.

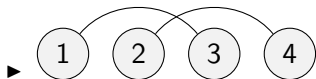
## Patterns on 4 vertices

Unfortunately it seems that the 3 vertices case is a very particular one.

The landscape is very different for sets of patterns on 4 vertices we have polynomial (such as outerplanar) but also NP-complete classes such as 3-colored graphs.

Can we understand the boarder in terms of patterns ?

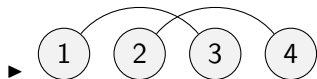
## On the polynomial side



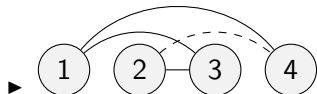
Outerplanar graphs



## On the polynomial side

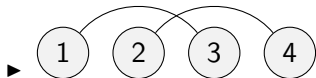


Outerplanar graphs

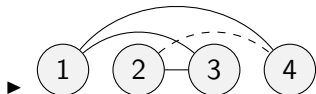


Strongly chordal graphs

## On the polynomial side



Outerplanar graphs

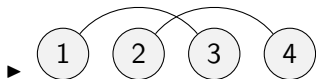


Strongly chordal graphs

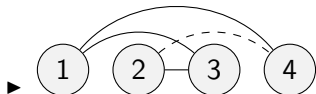
▶ patterns :  $K_4$  and  $\overline{K_4}$ .

Polynomial to recognize using  $Ramsey(4, 4) = 18$ . True for any fixed size.

## On the polynomial side



Outerplanar graphs



Strongly chordal graphs

- ▶ patterns :  $K_4$  and  $\overline{K_4}$ .

Polynomial to recognize using  $Ramsey(4, 4) = 18$ . True for any fixed size.

- ▶  $K_4$ -free graphs in  $O(m^\alpha)$ . Same for  $\overline{K_4}$ -free graphs, also true for any fixed size.

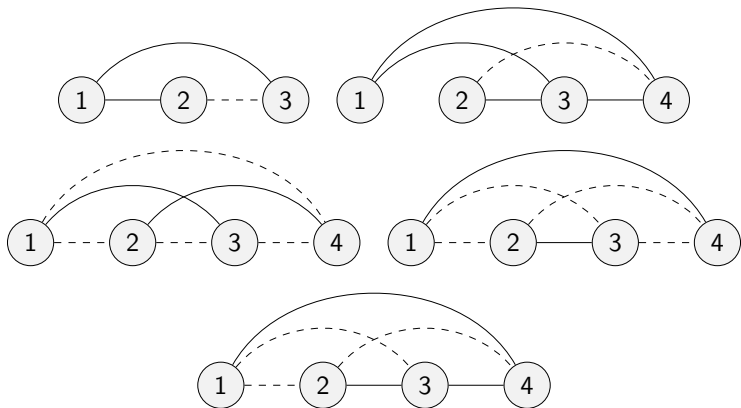
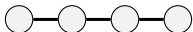


Figure: Forbidden patterns of coTT (Complement of Threshold Tolerance graphs).

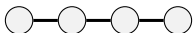
## On the NP-complete side

- ▶ 3 colorable graphs

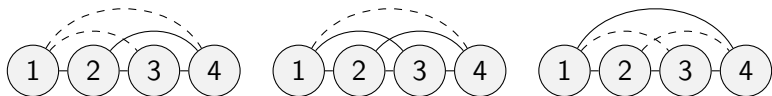


# On the NP-complete side

- ▶ 3 colorable graphs

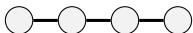


- ▶ perfectly orderable graphs

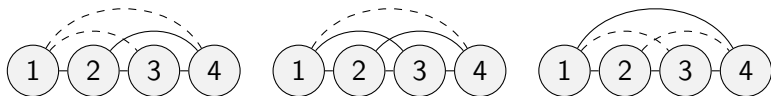


# On the NP-complete side

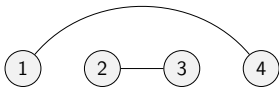
- ▶ 3 colorable graphs



- ▶ perfectly orderable graphs



- ▶ Forbidden pattern of the graphs of queue number 1.



Introduction

Patterns on 3 nodes

Patterns on 4 nodes

**Geometric classes**

Finding an ordering versus certifying it

Some Algorithmic Applications

Conclusions and perspectives



# We restrict our attention on patterns with 4 vertices that have a geometric flavor

Grounded intersection of objects



Figure: A grounded intersection model

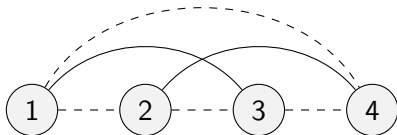
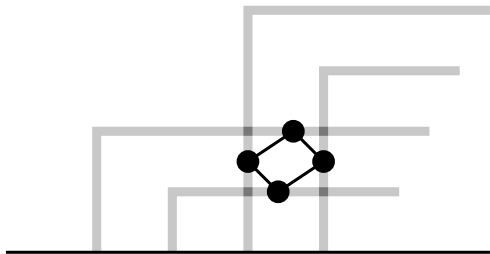
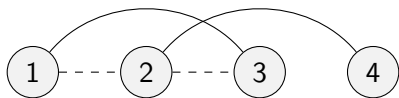


Figure: A pattern on four nodes

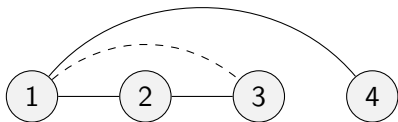
Figure: A grounded intersection model and a related pattern on four nodes.



Grounded L-shapes and the associated grounded-L graph.



Forbidden patterns



For grounded L-graphs.



(a) Grounded.



(b) Touching grounded.



(c) Outer.



(d) Bigrounded.

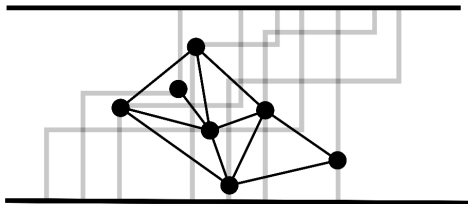
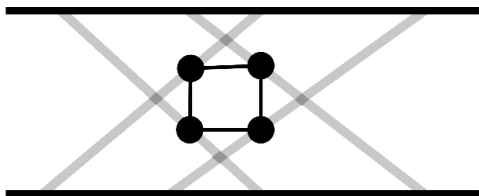


(e) 2-grounded.



(f) Circle.

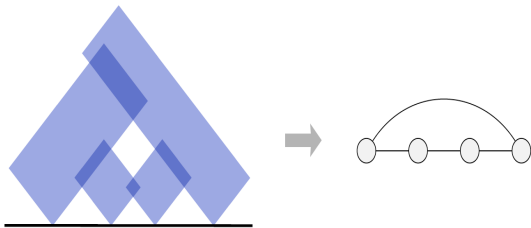
Permutation graph.



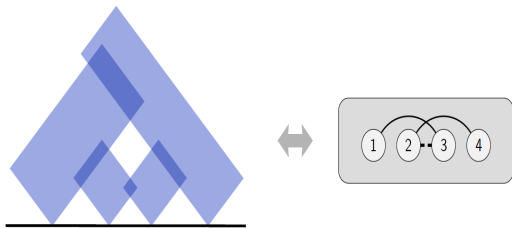
Co-comparability graph.

The permutation graph is a 4-cycle, that is not an interval graph. The graph that illustrates the cocomparability graphs is not a comparability graph, thus not a permutation graph.

# Grounded rectangles graphs



# Grounded rectangles graphs



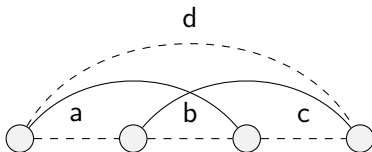
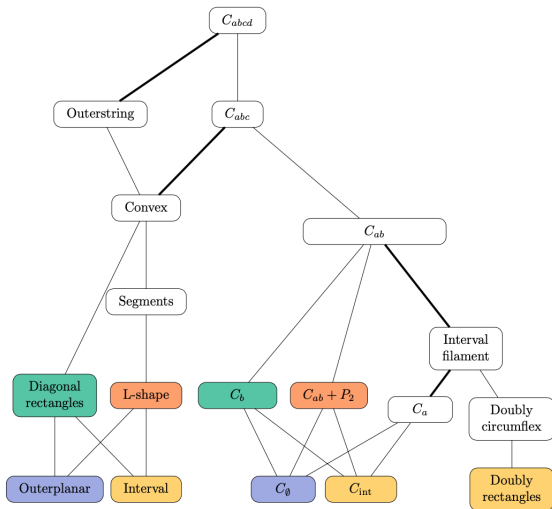


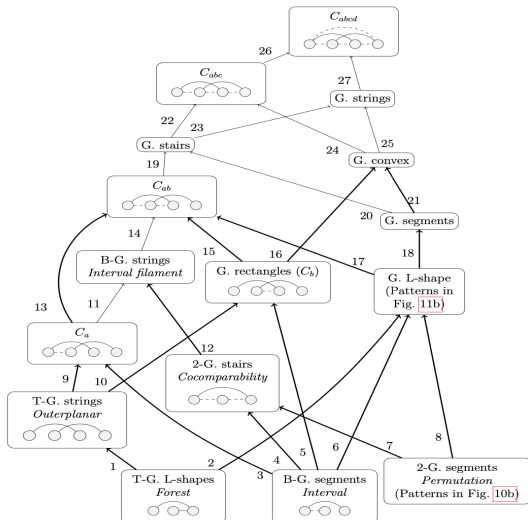
Figure: Description of  $abcd$  patterns.

We will consider the patterns that can be formed by having some subset of the non-edges. For  $S$  a subset of  $\{a, b, c, d\}$ , the pattern  $P_S$  is the pattern with the two plain edges of the above Figure, and the subset  $S$  of non-edges. The class  $C_S$  is the class associated with the pattern  $P_S$ .

$C_\emptyset$  = Outerplanar graphs,  $G_b$  = Gounded Rectangle graphs (Pbox-graphs).







**Diagram of our results.** The name of the classes in our vocabulary are written in normal font, the names in the literature are written in italic font. Edges represent inclusions. Thick edges are known strict inclusions. G. stands for *grounded*, T-G stands for *touching grounded*, B-G. stands for *Bigrounded*, and 2-G. stands for *2-grounded*.

Our results : a classification of these geometric graph classes using patterns.

1. The difficult part: proving strict inclusion between 2 classes, we need to understand:

If a pattern  $P$  is strictly included in a pattern  $Q$  as signed directed path graph under which conditions:  $\mathcal{C}_Q \subsetneq \mathcal{C}_P$  ?

Our proofs in the hierarchy were obtained by hand (or by computer ...)

Our results : a classification of these geometric graph classes using patterns.

1. The difficult part: proving strict inclusion between 2 classes, we need to understand:

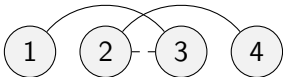
If a pattern  $P$  is strictly included in a pattern  $Q$  as signed directed path graph under which conditions:  $\mathcal{C}_Q \subsetneq \mathcal{C}_P$  ?

Our proofs in the hierarchy were obtained by hand (or by computer ...)

2. Characterizations as graph classes of  $\mathcal{C}_a$ ,  $\mathcal{C}_{ab}$ ,  $\mathcal{C}_{abc}$  and  $\mathcal{C}_{abcd}$  are still missing.

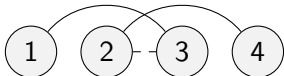
## Recent Results by D. Chakraborty, K. Gajjar and I. Rusu (ArXiv 2022)

- Recognition of Grounded Rectangle graphs is now known to be on the NP-complete side

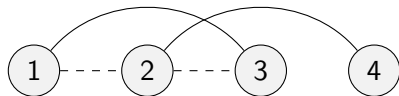


## Recent Results by D. Chakraborty, K. Gajjar and I. Rusu (ArXiv 2022)

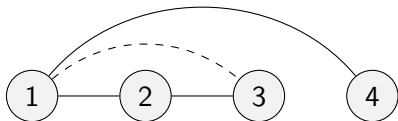
- ▶ Recognition of Grounded Rectangle graphs is now known to be on the NP-complete side



- ▶ Recognition of Grounded L-graphs also NP-complete 2022



Forbidden patterns



For grounded L-graphs.

Introduction

Patterns on 3 nodes

Patterns on 4 nodes

Geometric classes

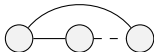
**Finding an ordering versus certifying it**

Some Algorithmic Applications

Conclusions and perspectives

Forbidden patterns and excluded subgraphs are complementary approaches for hereditary classes of graphs.

- ▶  $G$  chordal iff it admits a simplicial elimination scheme

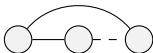


iff it does not contain a subgraph isomorphic to a chordless cycle of length  $\geq 4$ .



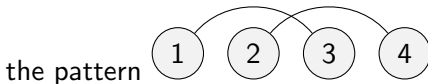
Forbidden patterns and excluded subgraphs are complementary approaches for hereditary classes of graphs.

- ▶  $G$  chordal iff it admits a simplicial elimination scheme



iff it does not contain a subgraph isomorphic to a chordless cycle of length  $\geq 4$ .

- ▶  $G$  outerplanar iff it admits an ordering of the vertices avoiding



iff it does not contain a cycle with two crossing edges.

- ▶ The ordering provides a certificate in the YES-case

- ▶ The ordering provides a certificate in the YES-case
- ▶ An excluded subgraph provides a certificate in the NO-case

- ▶ The ordering provides a certificate in the YES-case
- ▶ An excluded subgraph provides a certificate in the NO-case
- ▶ For  $K_p$  and its complement the two viewpoints are the same.

## For simple patterns on three vertices

- ▶ Except for Triangle-free, Comparability and their complement the ordering can be checked in linear time.

## For simple patterns on three vertices

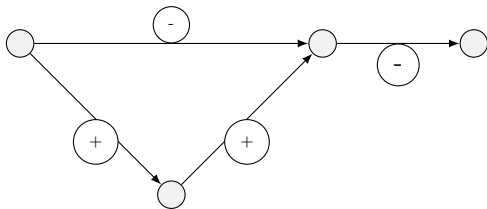
- ▶ Except for Triangle-free, Comparability and their complement the ordering can be checked in linear time.
- ▶ In fact, R. McConnell and J. Spinrad (1995) proposed a linear time algorithm to compute a comparability ordering (using modular decomposition).

## For simple patterns on three vertices

- ▶ Except for Triangle-free, Comparability and their complement the ordering can be checked in linear time.
- ▶ In fact, R. McConnell and J. Spinrad (1995) proposed a linear time algorithm to compute a comparability ordering (using modular decomposition).
- ▶ But to certify this ordering, no linear time algorithm is known (related to boolean matrix multiplication problem).

## Another interesting way to see our patterns

A pattern is an acyclic directed signed graph.



+ = a forced edge

- = a forced non edge

no arc = no constraint

Exchanging + and - corresponds to exchange  $\mathcal{C}$  to  $\text{co-}\mathcal{C}$ .



It allows to easily express what makes difficult the checking that an ordering avoids a given pattern.

Work in progress with François Pitois (IRIS Lyon) and Guillaume Ducoffe (Bucarest).

We want to characterize which patterns can be recognized in linear time. Using algorithms and reductions.

# As an appetizer

## Recognition of a simplicial elimination scheme in linear time

**Data:** A graph  $G$  given by its adjacency lists, and  $\sigma = x_1, \dots, x_n$   
a total ordering of  $V(G)$

**Result:** Yes :  $\sigma$  is a simplicial elimination scheme

No: a bad triple.

1  $\sigma$  simplicial iff  $\forall i, 1 \leq i \leq n$   $G(N(x_i) \cap \{x_{i+1}, \dots, x_n\})$  is a clique.

# The naive approach

---

---

1 **Simpliciality;**

**Data:** A graph  $G$  given by its adjacency lists, and  $\sigma = x_1, \dots, x_n$   
a total ordering of  $V(G)$

**Result:** Yes : $\sigma$  is a simplicial elimination scheme

No: a bad triple.

2 **for**  $i \leftarrow 1$  **to**  $n$  **do**

3 | Check if  $G(N(x_i) \cap \{x_{i+1}, \dots, x_n\})$  is a clique

---

## Complexity

Unfortunately this algorithm is not in linear time. If  $G$  itself is a clique, it works in  $O(n^3)$ . The worst case complexity is  $O(nm)$ .

---

---

1 **Simpliciality inspired from Tarjan and Yannakakis;**

**Data:** A graph  $G$  given by its adjacency lists, and  $\sigma = x_1, \dots, x_n$   
a total ordering of  $V(G)$

**Result:** Yes :  $\sigma$  is a simplicial elimination scheme

No: a bad triple.

2 Order the adjacency lists according to  $\sigma$  increasing;

3 **for**  $i \leftarrow 1$  **to**  $n$  **do**

4 |  $L(i) \leftarrow N(x_i) \cap \{x_{i+1}, \dots, x_n\};$

5 | **if**  $L(i)$  contains all lists attached to  $x_i$  **then**

6 | |  $x_k \leftarrow \text{First}(L(i));$  Extract  $x_k$  from  $L(i);$

7 | | Attach  $L(i)$  to  $x_k$

8 | **else**

9 | | Extract a bad triple and STOP;

---

# Proof

Main invariant :

$\forall i, 1 \leq i \leq n$ , after vertex  $x_i$  is processed,

$\forall j < i, N(x_j) \cap \{x_{j+1}, \dots, x_i\}$  is a clique

- ▶ Sorting the adjacency lists according to  $\sigma$ , can be done in  $O(|V(G)| + |E(G)|)$ .

- ▶ Sorting the adjacency lists according to  $\sigma$ , can be done in  $O(|V(G)| + |E(G)|)$ .
- ▶ For every vertex  $x_i$  its adjacency list is visited at most twice, first time to construct the list  $L(i)$  and a second time when checking its first neighbour according to  $\sigma$ .

- ▶ Sorting the adjacency lists according to  $\sigma$ , can be done in  $O(|V(G)| + |E(G)|)$ .
- ▶ For every vertex  $x_i$  its adjacency list is visited at most twice, first time to construct the list  $L(i)$  and a second time when checking its first neighbour according to  $\sigma$ .
- ▶ The comparison between the ordered lists attached to  $x_i$  can be done in  $|L(i)| +$  the size of the lists attached to  $x_i$ .



- ▶ Sorting the adjacency lists according to  $\sigma$ , can be done in  $O(|V(G)| + |E(G)|)$ .
- ▶ For every vertex  $x_i$  its adjacency list is visited at most twice, first time to construct the list  $L(i)$  and a second time when checking its first neighbour according to  $\sigma$ .
- ▶ The comparison between the ordered lists attached to  $x_i$  can be done in  $|L(i)| +$  the size of the lists attached to  $x_i$ .
- ▶ So the whole complexity is  $O(|V(G)| + |E(G)|)$ .

In case of failure : some list  $L(i)$  attached to a vertex  $x_j$  is not contained in  $L(j)$ . Therefore it exists  $x_h \in L(i) - L(j)$ , and we have a bad triple:  $x_i, x_h, x_j$ .

First linear time algorithm is from Tarjan Yannakakis

Remark: The algorithm is linear even if the input graph is not chordal.

## Algorithmic paradigm :

- ▶ Pass the work to your right neighbour

## Algorithmic paradigm :

- ▶ Pass the work to your right neighbour
- ▶ Can it be used to other elimination scheme ?

Introduction

Patterns on 3 nodes

Patterns on 4 nodes

Geometric classes

Finding an ordering versus certifying it

**Some Algorithmic Applications**

Conclusions and perspectives

We succeed with Lalla Mouatadib (2017) to put all known classes for which the induced maximum matching is polynomial in one unique statement using patterns.

# **Applications to distributed decision**

## A distributed NP

1. A prover gives to each node a small certificate
2. Every node gathers some  $t$ -neighbourhood (structure and certificates) and chooses to accept or reject.
3. The graph is accept iff all nodes accept.



## A distributed NP

1. A prover gives to each node a small certificate
2. Every node gathers some  $t$ -neighbourhood (structure and certificates) and chooses to accept or reject.
3. The graph is accept iff all nodes accept.

# Distributed NP recognition

The ordering is a useful certificate

that can be checked **locally** for many classes when the pattern is small.

This has been widely studied by L. Feuilloley and P. Fraigniaud.

- ▶ Question: comment l'ordre global est-il calculé ?

- ▶ Question: comment l'ordre global est-il calculé ?
- ▶ La réponse de Laurent F. : Pour le distribué dans le modèle de la certification locale, on ne se pose pas la question du calcul des certificats, on se demande juste à quel point on a besoin de gros certificats pour certifier telle ou telle propriété. Après le modèle vient de l'autostabilisation, où (dans certains cas) on peut faire plein de rondes de communication (donc en particulier simuler des parcours dans tous les sens), du moment que l'on n'utilise pas trop de mémoire à chaque sommet.

Introduction

Patterns on 3 nodes

Patterns on 4 nodes

Geometric classes

Finding an ordering versus certifying it

Some Algorithmic Applications

**Conclusions and perspectives**

## What else is known?

- ▶ Many recognitions are **NP-complete**, e.g.  $k$ -colourability (pattern = a path of length  $k$ );

## What else is known?

- ▶ Many recognitions are **NP-complete**, e.g.  $k$ -colourability (pattern = a path of length  $k$ );
- ▶ Almost all the classes defined by 2-connected patterns are NP-complete to recognize [Duffus, Ginn, Rodl, 1995].

## What else is known?

- ▶ Many recognitions are **NP-complete**, e.g.  $k$ -colourability (pattern = a path of length  $k$ );
- ▶ Almost all the classes defined by 2-connected patterns are NP-complete to recognize [Duffus, Ginn, Rodl, 1995].
- ▶ Hell, Mohar and Rafiey conjectured a dichotomy theorem.



## What else is known?

- ▶ Many recognitions are **NP-complete**, e.g.  $k$ -colourability (pattern = a path of length  $k$ );
- ▶ Almost all the classes defined by 2-connected patterns are NP-complete to recognize [Duffus, Ginn, Rodl, 1995].
- ▶ Hell, Mohar and Rafiey conjectured a dichotomy theorem.
- ▶ It seems that there is no such dichotomy [Nesetril 2017].

## S. Guzmán-Pro, P. Hell and C. Hernández-Cruz (Arxiv 2021)

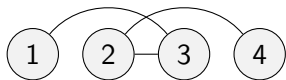
- ▶ Same results if we consider acyclic orientation avoiding a set of patterns. More precisely the graph classes are the same.

## S. Guzmán-Pro, P. Hell and C. Hernández-Cruz (Arxiv 2021)

- ▶ Same results if we consider acyclic orientation avoiding a set of patterns. More precisely the graph classes are the same.
- ▶ Some work on circular patterns.

## But there are still many open problems

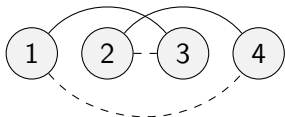
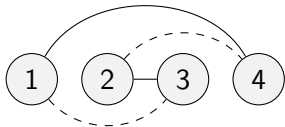
What is the status of this pattern?



Could be NP-complete as well as for the 12 other drawings of a  $P_4$ ?

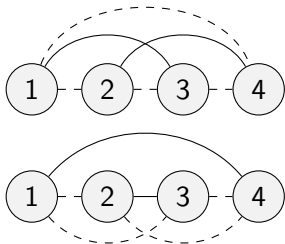
A question from Santiago Guzman: are they connected to some kind of coloring?

And these two?



Which correspond to linear Hsu decomposable graphs or mimwidth 1 (cf. J.A. Telle).

## Similarly



Which correspond to linear simwidth 1 graphs (cf. J.A. Telle). The first pattern is  $P_{\{a,b,c,d\}}$ .

## Back to our initial question

Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?

- ▶ Not completely answered!

## Back to our initial question

Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?

- ▶ Not completely answered!
- ▶ Could be something related with geometric discrete convexity and graph searches (cf. the cograph case).



## Back to our initial question

Why the linear-time recognition algorithms based on LexBFS for chordal, proper interval, interval and cocomparability graphs produce an ordering that avoids their characteristic pattern?

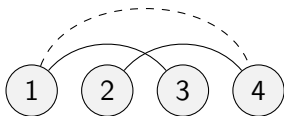
- ▶ Not completely answered!
- ▶ Could be something related with geometric discrete convexity and graph searches (cf. the cograph case).
- ▶ But it is the subject of another lecture (next 15<sup>th</sup> in Montpellier) and not finished.

## Some extensions

- ▶ A better understanding of the landscape for patterns of size 4.

## Some extensions

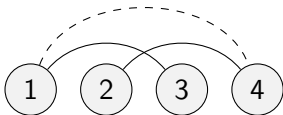
- ▶ A better understanding of the landscape for patterns of size 4.
- ▶ Several nice patterns coming from geometric objects to study, such as  $P_d$ .



Not yet properly inserted in our hierarchy!

## Some extensions

- ▶ A better understanding of the landscape for patterns of size 4.
- ▶ Several nice patterns coming from geometric objects to study, such as  $P_d$ .



Not yet properly inserted in our hierarchy!

- ▶ Some rule to extend a pattern by adding a vertex (as for the path).

## Other interesting extensions

- ▶ Extension to patterns on 3 nodes with directed cycles, in order to capture other graph classes (cf. Reza Naserasr?)

## Other interesting extensions

- ▶ Extension to patterns on 3 nodes with directed cycles, in order to capture other graph classes (cf. Reza Naserasr?)
- ▶ Generalizations to temporal graphs . . . (cf. Laurent Viennot?)

## Other interesting extensions

- ▶ Extension to patterns on 3 nodes with directed cycles, in order to capture other graph classes (cf. Reza Naserasr?)
- ▶ **Generalizations to temporal graphs . . . (cf. Laurent Viennot?)**
- ▶ Some work already done on temporal extensions of comparability graphs by Mertzios, Molter, Renken, Spirakis and Zschoche ( ArXiv 2021)

## Other interesting extensions

- ▶ Extension to patterns on 3 nodes with directed cycles, in order to capture other graph classes (cf. Reza Naserasr?)
  - ▶ Generalizations to temporal graphs . . . (cf. Laurent Viennot?)
  - ▶ Some work already done on temporal extensions of comparability graphs by Mertzios, Molter, Renken, Spirakis and Zschoche ( ArXiv 2021)
- 
- ▶ Many thanks for your attention !



## Some References



José R. Correa, Laurent Feuilloley, Pablo Pérez-Lantero, and José A. Soto.  
Independent and hitting sets of rectangles intersecting a diagonal line: Algorithms and complexity.  
*Discrete & Computational Geometry*, 53(2):344–365, 2015.



Peter Damaschke.  
Forbidden ordered subgraphs.  
*Topics in Combinatorics and Graph Theory*, R. Bodendiek and R. Henning Eds, pages 219–229, 1990.



Dwight Duffus, Mark Ginn, and Vojtech Rödl.  
On the computational complexity of ordered subgraph recognition.  
*Random Struct. Algorithms*, 7(3):223–268, 1995.



Laurent Feuilloley and Michel Habib.  
Classifying grounded intersection graphs via ordered forbidden patterns.  
*CoRR*, abs/2112.00629, 2021.



Laurent Feuilloley and Michel Habib.  
Graph classes and forbidden patterns on three vertices.  
*SIAM J. Discret. Math.*, 35(1):55–90, 2021.



Santiago Guzmán-Pro, Pavol Hell, and César Hernández-Cruz.  
Describing hereditary properties by forbidden circular orderings.  
*CoRR*, abs/2112.00154, 2021.



Pavol Hell, Bojan Mohar, and Arash Rafiey.  
Ordering without forbidden patterns.  
*In Algorithms - ESA 2014 - 22th Annual European Symposium*, pages 554–565, 2014.



George B. Mertzios, Hendrik Molter, Malte Renken, Paul G. Spirakis, and Philipp Zschoche.  
The complexity of transitively orienting temporal graphs.  
*CoRR*, abs/2102.06783, 2021.