A structure theorem for graphs with no cycle with a unique chord and its consequences

Sophia Antiplolis — November 2008

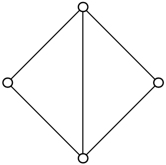
Nicolas Trotignon

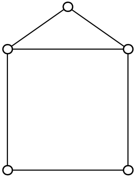
CNRS — LIAFA — Université Paris 7

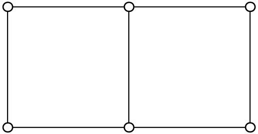
Joint work with:

- Kristina Vušković
  School of Computing,
  University of Leeds.

Cycles with a unique chord







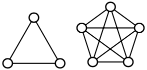
Our problem

- Study the **structure** of:
  graphs that do not contain
  a cycle with a unique chord
  as an induced subgraph

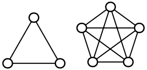
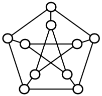
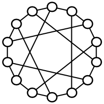
- Notation :
  $\mathcal{C}$ = **class of these graphs**

Every graph in $\mathcal{C}$ either:

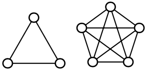
- is **basic**
- has a **decomposition**

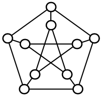
- **cliques**: 

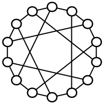
Basic classes

- **cliques**: 
- induced subgraphs of the **Petersen graph**: 
- induced subgraphs of the **Heawood graph**: 

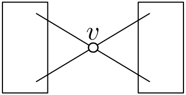
Basic classes

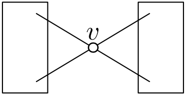
- **cliques**: 

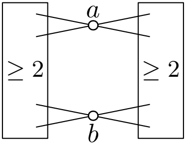
- induced subgraphs of the **Petersen graph**: 

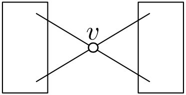
- induced subgraphs of the **Heawood graph**: 

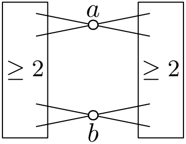
- **strongly 2-bipartite graphs** : graph that are bipartite and one side contains only vertices of degree 2.

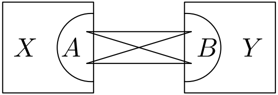
- **1-cutset**: 

- **1-cutset**: 

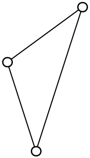
- **2-cutset**: 

- **1-cutset**: 

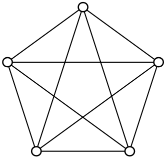
- **2-cutset**: 

- **1-join**: 

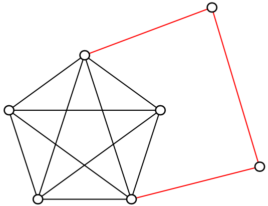
Proof: case triangle



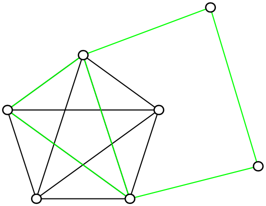
Proof: case triangle



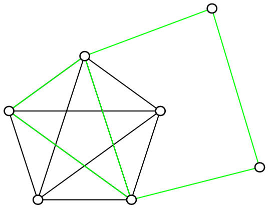
Proof: case triangle



Proof: case triangle

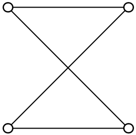


Proof: case triangle

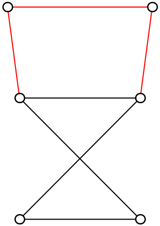


So: **triangle → clique or 1-cutset**

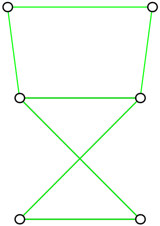
Proof: case square



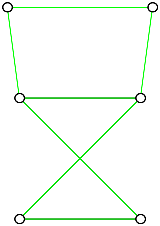
Proof: case square



Proof: case square

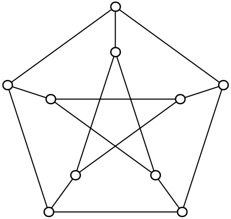


Proof: case square

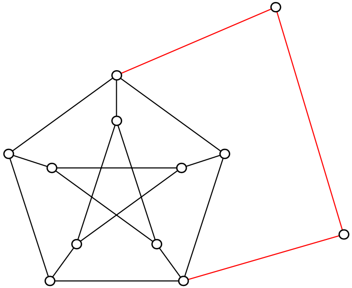


So: **square → 1-join**

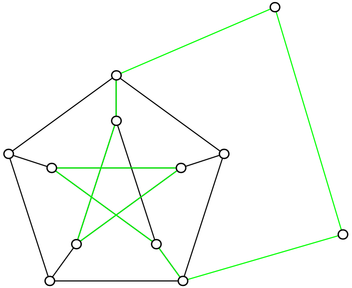
Proof: case Petersen

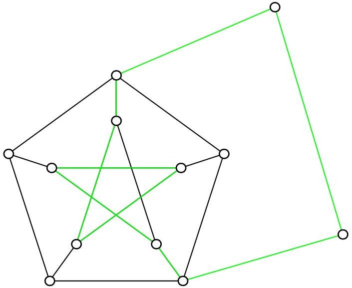


Proof: case Petersen



Proof: case Petersen

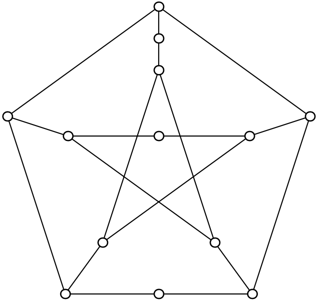




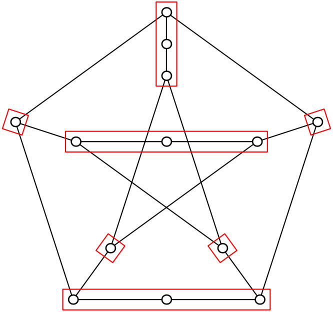
So: **Petersen → Petersen or 1-cutset**

Similarly: **Heawood**

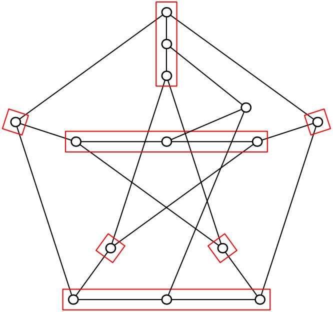
Proof: case 3 paths “like that”



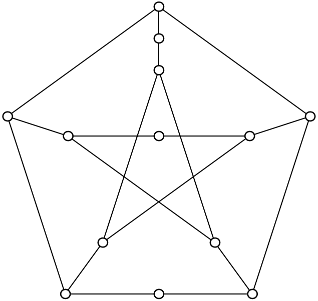
Proof: case 3 paths “like that”



Proof: case 3 paths “like that”



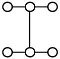
Proof: case 3 paths “like that”



So: **3 paths like that →**

Heawood minus one vertex or 1-cutset, or 2-cutset

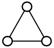
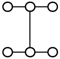
Proof: a lot of cases go that way

- After eliminating a dozen of configurations we can prove:
- If the graph contains: 

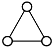
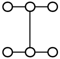
Then the graph is **basic or has a decomposition**

Proof: the end

The graph may now be assumed to contain no:

- 
- 
- 

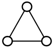
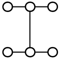
The graph may now be assumed to contain no:

- 
- 
- 

So: no 2 vertices of degree ≥ 3 are adjacent

Proof: the end

The graph may now be assumed to contain no:

- 
- 
- 

So: no 2 vertices of degree ≥ 3 are adjacent

Hence, the graph is **strongly 2-bipartite** or has a **2-cutset**

- **Structural description**

Motivation 1

- **Structural description**
- Our decompositions are **reversible**

Motivation 1

- **Structural description**
- Our decompositions are **reversible**
- This is **algorithmic**. For every graph in $\mathcal{C}$ we build a decomposition tree in time $O(nm)$

- **Structural description**
- Our decompositions are **reversible**
- This is **algorithmic**. For every graph in $\mathcal{C}$ we build a decomposition tree in time $O(nm)$
- We use involved subroutines for finding decompositions in linear time, due to:
  - **Hopcroft and Tarjan for 1-cutsets and 2-cutsets**
  - **Dahlhaus for 1-joins**

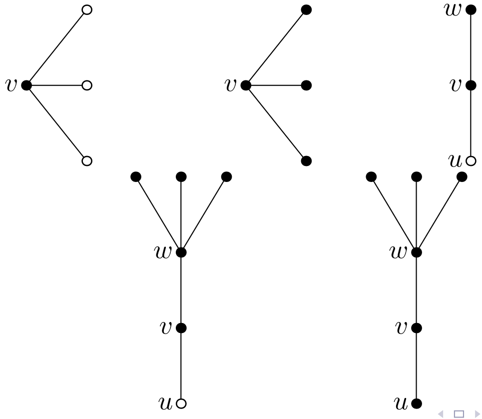
Motivation 2

- Properties of graph invariants:
  - For very graph G in $\mathcal{C}$:

$$\chi(G) = 3 \textbf{ or } \chi(G) = \omega(G)$$

- Algorithms:
  - $O(nm)$ for coloring
  - $O(n+m)$ for maximum clique
  - Maximum stable set is NP-hard [Poljak, 1974]

- Every triangle-free graph of $\mathcal{C}$ is 3-colorable.
  Proved by **induction**.
- The plain induction does not work.
  A coloring with **constraints** needs to be done:



- **Detection of induced subgraphs**

- We have an $O(nm)$-**time algorithm** that detects cycles with a unique chord.

A problem that is too difficult

- Instance: two graphs, G and H
- Question: is H an induced subgraph of G ?

A problem that is too difficult

- Instance: two graphs, G and H
- Question: is H an induced subgraph of G ?

This problem is NP-complete [Cook, 1971].

A problem that is too easy

Let H be a graph, and let us consider the problem:

- Instance: one graph G
- Question: is H an induced subgraph of G ?

A problem that is too easy

Let H be a graph, and let us consider the problem:

- Instance: one graph G
- Question: is H an induced subgraph of G ?

This problem is polynomial (trivial by a brute-force search).

A problem that might be easy or difficult

Let $\mathcal{H}$ be a set of graphs.

- Instance: one graph G
- Question: is there any graph $H \in \mathcal{H}$ such that H is an induced subgraph of G ?

A problem that might be easy or difficult

Let $\mathcal{H}$ be a set of graphs.

- Instance: one graph G
- Question: is there any graph $H \in \mathcal{H}$ such that H is an induced subgraph of G ?

This problem is polynomial when $\mathcal{H}$ is finite.
When $\mathcal{H}$ is infinite, the problem can be polynomial, NP-complete, or most of the time open ...

An **s-graph** is a graph with two kinds of edges:

An **s-graph** is a graph with two kinds of edges:

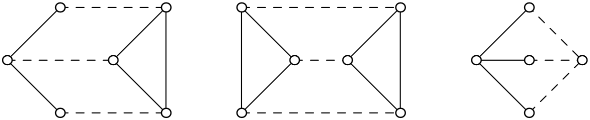
- edges

An **s-graph** is a graph with two kinds of edges:

- edges
- subdivisible edges

An **s-graph** is a graph with two kinds of edges:

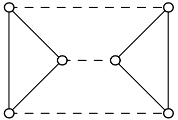
- edges
- subdivisible edges



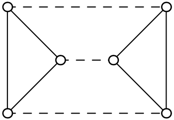
Realisation of an s-graph

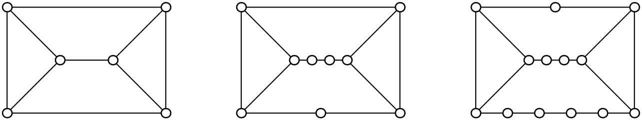
A **realisation** of an s-graph is a graph obtained by subdividing subdivisible edges of the s-graph.

A **realisation** of an s-graph is a graph obtained by subdividing subdivisible edges of the s-graph.



A **realisation** of an s-graph is a graph obtained by subdividing subdivisible edges of the s-graph.

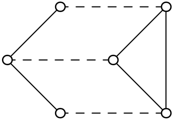
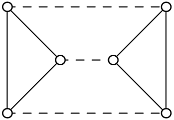
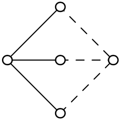




Given an s-graph H, we consider the problem Π_H:

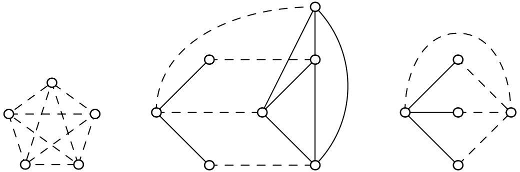
- Instance: A graph G
- Question: Does G contain any realisation of H as an induced subgraph ?

Important examples: initial motivation

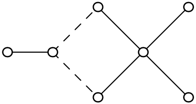
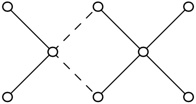
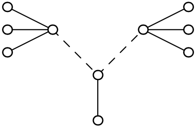
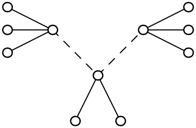
- 
  Polynomial, $O(n^9)$,
  Chudnovsky and Seymour, 2002
- 
  NP-complete,
  Maffray and N.T., 2003
- 
  Polynomial, $O(n^{11})$,
  Chudnovsky and Seymour, 2006

Other examples of interest:

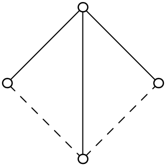
In joint work with Lévêque, Lin and Maffray, we proved that the following problems are NP-complete:



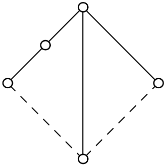
We prove (with Lévêque, Lin and Maffray):

-  Polynomial, $O(n^{13})$
-  NP-complete
-  Polynomial, $O(n^{14})$
-  NP-complete

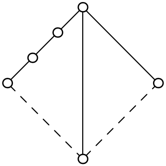
Other stricking examples



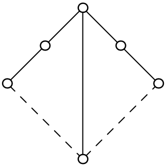
$H_{1|1}$:



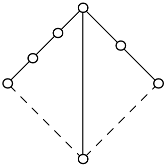
$H_{2|1}$:



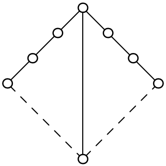
$H_{3|1}$:



$H_{2|2}$:

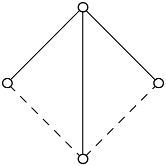


$H_{3|2}$:

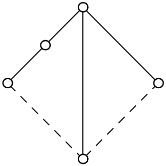


$H_{3|3}$:

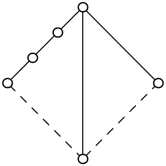
Other stricking examples



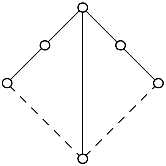
$H_{1|1}$: $O(nm)$



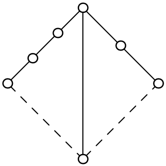
$H_{2|1}$:



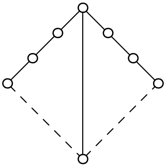
$H_{3|1}$:



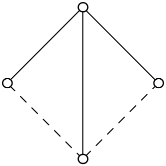
$H_{2|2}$:



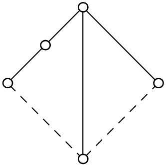
$H_{3|2}$:



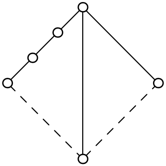
$H_{3|3}$:



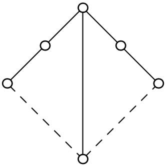
$H_{1|1}$: $O(nm)$



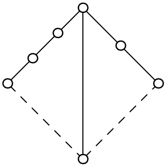
$H_{2|1}$: ?



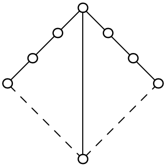
$H_{3|1}$: ?



$H_{2|2}$: ?

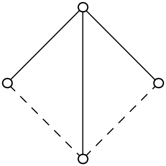


$H_{3|2}$: ?

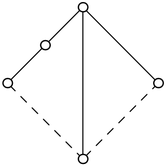


$H_{3|3}$:

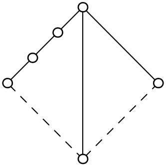
Other stricking examples



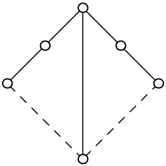
$H_{1|1}$: $O(nm)$



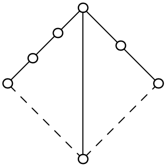
$H_{2|1}$: ?



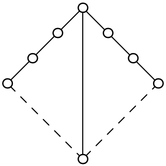
$H_{3|1}$: ?



$H_{2|2}$: ?



$H_{3|2}$: ?



$H_{3|3}$: **NPC**

three-in-a-tree:

- Instance: A graph G and three vertices a, b, c of G
- Question: Is there an induced tree going through a, b, c ?

Can be solved in $O(n^4)$, Chudnovsky and Seymour 2006

three-in-a-tree:

- Instance: A graph G and three vertices a, b, c of G
- Question: Is there an induced tree going through a, b, c ?

Can be solved in $O(n^4)$, Chudnovsky and Seymour 2006

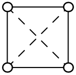
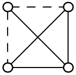
All the polynomial algorithms mentioned above are done (or can be done) by using three-in-a-tree.

One exception: detecting a cycle with a unique chord

Survey of complexity for s-graphs on 4 vertices

For the following two s-graphs, there is a **polynomial** algorithm using three-in-a-tree:

The next two s-graphs yield an **NP-complete** problem:

 (by $\Pi_{\{C_4\}}$)  (by $\Pi_{\{K_3\}}$)

For the remaining eight ones, we **do not know** the answer: