

# Algorithmic aspects of modular decomposition

## Cours MPRI 2011–2012

Michel Habib

habib@liafa.jussieu.fr

<http://www.liafa.jussieu.fr/~habib>

Chevaleret november 2011

# Schedule of the talk

## Basic Definitions on Modules

Partitive Families

## Modular Decomposition Algorithms

Historical Notes

Bottom up Techniques

## Transitive orientations and modular decomposition

Top Down techniques

Partition refinement techniques

Cographs

### Joint work with :

Binh Minh Bui Xuan (Univ. P. et M. Curie, CR CNRS)

Alain Cournier (Univ. de Picardie Jules Verne Amiens, Pr)

Vincent Limouzy (Limos, Clermont-Ferrand, MdC)

Fabien de Montgolfier (Univ. Paris Diderot, LIAFA, MdC)

Ross McConnell (Colorado State University))

Christophe Paul (Lirmm, Montpellier DR CNRS)

Jerry Spinrad (Vanderbilt University, Nashville) ...

### Conventions

Graphs considered in the following are undirected, loopless.

$$G = (V, E)$$

$N_G(x)$  or simply  $N(x)$  the neighbourhood of  $x$  in  $G$ .

# Modules

## Modules

For a graph  $G = (V, E)$ , a **module** is a subset of vertices  $A \subseteq V$  such that

$$\forall x, y \in A, N(x) - A = N(y) - A$$

The problem with this definition : must we check all subsets  $A$ ?

## Trivial Modules

$\emptyset$ ,  $\{x\}$  and  $V$  are modules.

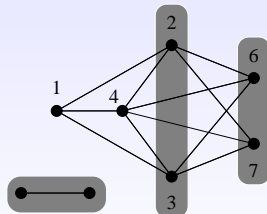
## Prime Graphs

A graph is **prime** if it admits only trivial modules.

## Examples

### Characterization of Modules

A subset of vertices  $M$  of a graph  $G = (V, E)$  is a **module** iff  
 $\forall x \in V \setminus M$ , either  $M \subseteq N(x)$  or  $M \cap N(x) = \emptyset$



### Examples of modules

- ▶ connected components of  $G$
- ▶ connected components of  $\overline{G}$
- ▶ any vertex subset of the complete graph (or the stable)

## Playing with the definition

### Duality

$A$  is a module of  $G$  implies  $A$  is a module of  $\overline{G}$ .

### Easy observations

- ▶ No prime graph with  $\leq 3$  vertices.
- ▶  $P_4$  the path with 4 vertices is the only prime on 4 vertices.
- ▶  $P_4$  is isomorphic to its complement.

## Particular modules : twins and strong modules

### Twins

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

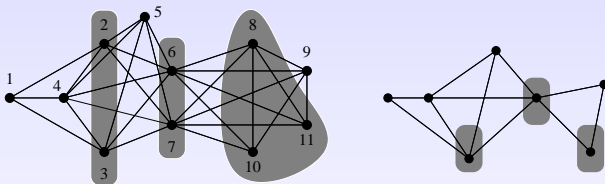
$x, y$  are false twins in  $G$  iff  $x, y$  are true twins in  $\overline{G}$ .

### Strong modules

A **strong module** is a module that does not strictly overlap any other module.

## Modular partition

A partition  $\mathcal{P}$  of the vertex set of a graph  $G = (V, E)$  is a **modular partition** of  $G$  if any part of  $\mathcal{P}$  is a module of  $G$ .

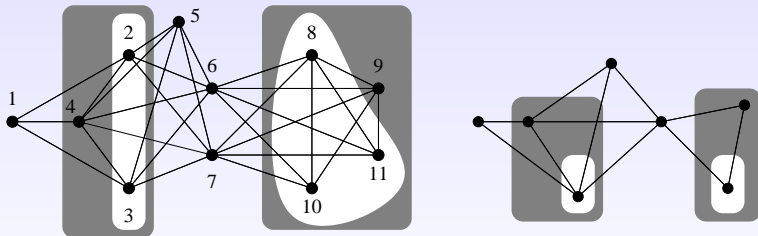


Let  $\mathcal{P}$  be a modular partition of a graph  $G = (V, E)$ . The **quotient graph**  $G/\mathcal{P}$  is the induced subgraph obtained by choosing one vertex per part of  $\mathcal{P}$ .

## Lemma (Mohring Radermacher 1984)

Let  $\mathcal{P}$  be a modular partition of  $G = (V, E)$ .

$\mathcal{X} \subseteq \mathcal{P}$  is a module of  $G_{/\mathcal{P}}$  iff  $\cup_{M \in \mathcal{X}} M$  is a module of  $G$ .

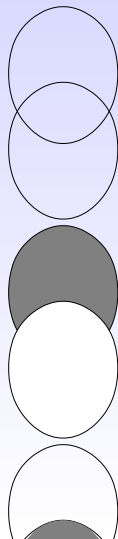


## Partitive Families

### Lemma

*If  $M$  and  $M'$  are two overlapping modules then*

- ▶ *(i)  $M \setminus M'$  is a module*
- ▶ *(ii)  $M \cap M'$  is a module*
- ▶ *(iii)  $M \cup M'$  is a module*
- ▶ *(iv)  $M \Delta M'$  is a module*



## Easy observation

The set of all modules of an undirected graph (resp. a directed graph) constitutes a partitive family (resp. a weak partitive family).

### Theorem Gallai 1967

Let  $G = (V, E)$  be a graph with  $|V| \geq 4$ , the three following cases are mutually exclusive :

1.  $G$  is not connected,
2.  $\overline{G}$  is not connected,
3.  $G/\mathcal{M}(G)$  is a prime graph, with  $\mathcal{M}(G)$  the modular partition containing the maximal strong modules of  $G$ .

## Proof of the Modular Decomposition Theorem

- ▶ If  $G$  is disconnected into  $k$  connected components  $G_1, G_2 \dots G_k$  then  $\overline{G}$  contains all the edges between the  $G_i$ 's and therefore is connected. Dual argument if  $\overline{G}$  is disconnected. Therefore the two first cases are exclusive.
- ▶ So now we consider a graph which is connected and co-connected (i.e. its complement is connected). Let us consider 2 maximal modules  $M, M'$  of  $G$  (under inclusion), we claim that they are disjoint. Else since they are maximal they necessarily overlap.  
But modules are partitive : necessarily  $M \cup M' = V$  and  $M$  and  $M' - M$  are modules which partition  $V$ . This contradicts  $G$  and  $\overline{G}$  being connected.
- ▶ To finish we consider the modular partition obtained with this maximal modules and necessarily the quotient graph is prime.

## Modular Decomposition Theorem for Directed Graphs

### Theorem

Let  $G = (V, E)$  be a directed graph with  $|V| \geq 3$ , the four following cases are mutually exclusive :

1.  $G$  is not connected, *Parallel node*
2.  $\overline{G}$  is not connected, *Series node*
3.  $G^*$  is not strongly connected, *Linear node*
4.  $G_{/\mathcal{M}(G)}$  is a prime graph, with  $\mathcal{M}(G)$  the modular partition containing the maximal strong modules of  $G$ , *Prime node*

The proof is similar for the directed case.

As a byproduct, we notice that an undirected prime graph  $G$  satisfies :

$G$  and  $\overline{G}$  are connected

## Modular decomposition tree for undirected graph

### Tree

Using recursively this theorem we obtain a unique tree  $T$  in which : the root corresponds to  $V$ , leaves are associated to vertices and each node corresponds to a strong module.

There are 3 types of nodes :

**Parallel**  $G$  disconnected and every union of connected components is a module of  $G$ . Strong modules are associated to these connected components.

**Series**  $\overline{G}$  is disconnected and every union of co-connected components is a module of  $G$ . Strong modules are associated to these co-connected components.

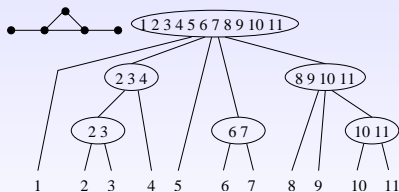
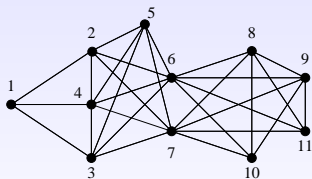
**Prime**  $G$  admits a unique partition of the vertex set in maximal modules (strong).

## Another formulation

### Strong Modules

The set of strong modules is nested into an inclusion tree (called the **modular decomposition tree**  $MD(G)$  of  $G$ ).

**Strong modules are the more interesting ones.**



**Problem :** find the best algorithm for computing this modular decomposition tree

## Uniqueness decomposition theorem for families Chein, Habib, Maurer 1981

Partitive (resp. weakly partitive) families admit a decomposition tree with two (resp. three) types of nodes :

- ▶ degenerate (also called fragile)
- ▶ prime
- ▶ (resp. linear)

This tree representation theorem for partitive (resp. weakly partitive) families  $F \subseteq 2^{|V|}$ , yields a tree encoding of these families in  $O(|V|)$ . It should be noticed that such a family may have an exponential number of elements but could be described with an  $O(|V|)$  tree.

$\overline{G}$

## Prime graphs are nested

### Folklore Theorem

Let  $G$  be a prime graph ( $|G| \geq 4$ ), then  $G$  contains a  $P_4$ .

### Theorem Schmerl, Trotter, Ille 1991 ...

Let  $G$  be a prime graph ( $|G| = n \geq 4$ ), then  $G$  contains a prime graph on  $n - 1$  vertices or a prime graph on  $n - 2$  vertices.

## A simple proof

A stronger statement, Cournier, Ille, 1991

For a prime graph there is at most one vertex not contained in a  $P_4$ .

### Proof

As a prime graph  $G$  is necessarily connected and if it  $\exists x \in V$  that does not belong to a  $P_4$  then  $\overline{N(x)}$  is a stable set.

If  $\exists x \neq y \in V$  that does not belong to a  $P_4$ ,

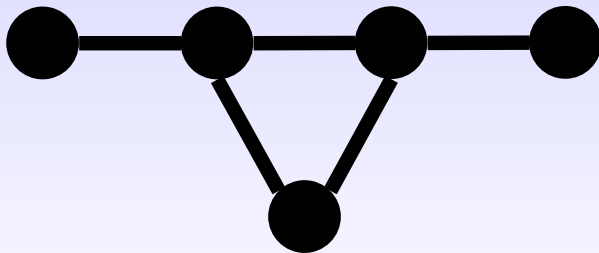
wlog assume  $\underline{xy \notin E}$

But then  $y \in \overline{N(x)}$  and therefore :  $N(y) \subseteq N(x)$ .

By symmetry  $x, y$  must be false twins, **a contradiction.**

Nota Bene : this result also holds for infinite graphs !  
So does our previous proof.

In fact if there is such a vertex  $x \in V$ ,  
 $x$  is adjacent to the middle vertices of a  $P_4$ .  
(Such a subgraph is called a **bull**).



A bull is isomorphic to its complement.

## Historical notes

The big list of published algorithms for modular decomposition (N.B. Perhaps some items are missing . . . please give me the missing references)

- ▶ Cowan, James, Stanton 1972  $O(n^4)$
- ▶ Maurer 1977  $O(n^4)$  directed graphs
- ▶ Blass 1978  $O(n^3)$
- ▶ Habib, Maurer 1979  $O(n^3)$
- ▶ Habib 1981  $O(n^3)$  directed graphs
- ▶ Corneil, Perl, Stewart 1981,  $O(n + m)$  cograph recognition.
- ▶ Cunningham 1982  $O(n^3)$  directed graphs
- ▶ Buer, Mohring 1983  $O(n^3)$
- ▶ McConnell 1987  $O(n^3)$
- ▶ McConnell, Spinrad 1989  $O(n^2)$  incremental
- ▶ Adhar, Peng 1990  $O(\log^2 n)$ ,  $O(nm)$  proc. parallel, cographs, CRCW-PRAM

- ▶ Lin, Olariu 1991  $O(\log n)$ ,  $O(nm)$  proc. parallel, cographs, EREW-PRAM
- ▶ Spinrad 1992  $O(n + \text{malpha}(m, n))$
- ▶ Cournier, Habib 1993  $O(n + \text{malpha}(m, n))$
- ▶ Ehrenfeucht, Gabow, McConnell, Spinrad 1994  $O(n^3)$   
2-structures
- ▶ Ehrenfeucht, Harju, Rozenberg 1994  $O(n^2)$  2-structures,  
incremental
- ▶ McConnell, Spinrad 1994  $O(n + m)$
- ▶ Cournier, Habib 1994  $O(n + m)$
- ▶ Bonizzoni, Della Vedova 1995  $O(n^{3k-5})$  Committee  
decomposition for hypergraphs
- ▶ Dahlhaus 1995  $O(\log^2 n)$ ,  $O(n + m)$  proc. parallel, cographs,  
CRCW-PRAM
- ▶ Dahlhaus 1995  $O(\log^2 n)$ ,  $O(n + m)$  proc. parallel,  
CRCW-PRAM

- ▶ Habib, Huchard, Spinrad 1995  $O(n + m)$  inheritance graphs
- ▶ McConnell 1995  $O(n^2)$  2-structures, incremental
- ▶ Capelle, Habib 1997  $O(n + m)$  if a factoring permutation is given
- ▶ Dahlhaus, Gustedt, McConnell 1997  $O(n + m)$
- ▶ Dahlhaus, Gustedt, McConnell 1999  $O(n + m)$  directed graphs
- ▶ Habib, Paul, Viennot 1999  $O(n + m \log n)$  via a factoring permutation
- ▶ McConnell, Spinrad 2000  $O(n + m \log n)$
- ▶ Habib, Paul 2001  $O(n + m)$  cographs via a factoring permutation
- ▶ Capelle, Habib, Montgolfier 2002  $O(n + m)$  directed graphs if a factoring permutation is provided.
- ▶ Shamir, Sharan 2003  $O(n + m)$  cographs, fully-dynamic
- ▶ McConnell, Montgolfier 2003  $O(n + m)$  directed graphs
- ▶ Habib, Montgolfier, Paul 2003  $O(n + m)$  computes a factoring permutation

- ▶ McConnell, de Montgolfier 2003  $O(n + m)$  Interval graphs and dimension  $k$  orders
- ▶ Brestcher, Corneil, Habib, Paul, 2006  $O(n + m)$  Cographs via LexBFS.
- ▶ ...
- ▶ But also, Uno, Yagura, 2000  $O(n)$  to decompose permutation graphs when a layout is provided.
- ▶ Bergeron, Chauve, de Montgolfier, Raffinot 2006
- ▶ **Simpler Linear-Time Modular Decomposition via Recursive Factorizing Permutation** Tedder, Corneil, Habib, Paul, ICALP (1) 2008 : 634-646.

## Why it is so important ?

[Jerry Spinrad' book 03]

The new [linear time] algorithm [MS99] is currently too complex to describe easily [...]. The first  $O(n^2)$  partitioning algorithms were similarly complex ; I hope and believe that in a number of years the linear algorithm can be simplified as well.

## Applications of modular decomposition

- ▶ A very natural operation to define on discrete structures, searching regularities.
- ▶ A structure theory for comparability graphs, close relationship with transitive orientation
- ▶ A compact encoding using module contraction and if we keep at each prime node the structure of the prime graph.
- ▶ Divide and conquer paradigm can be applied to solve optimization problems. For example to test isomorphism.

- ▶ A very basic graph algorithmic problem (similar to graph isomorphism problem).
- ▶ A better understanding of graph algorithms and their data structures.

## Minimal Modules

### Minimal module containing a set

For every  $A \subseteq V$  there exists a unique minimal module containing  $A$

### Proof :

Since the module family is partitive and therefore closed under  $\cap$ .

# Splitters

## Definition

A splitter for a  $A \subseteq V$ , is a vertex  $z \notin A$   
s.t.  $\exists x, y \in A$  with  $zx \in E$  and  $zy \notin E$ .

## Modules

$A \subseteq V$  is a module iff  $A$  does not have any splitter.

## Usefull lemma

If  $z$  is a splitter for a  $A \subseteq V$ , then any module containing  $A$  must also contain  $z$ .

## Submodularity

Let us denote by  $s(A)$  the number of splitters of a set  $A$ , then  $s$  is a submodular function.

### Definition

A function is submodular if

$$\forall A, B \subseteq E$$

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

This is the basis idea of Uno and Yagura's algorithm for the modular decomposition of permutation graphs in  $O(n)$ .

## Bottom-up Techniques

### Sketch of the algorithm

For each pair of vertices  $x, y \in V$

Compute the minimal module  $m(x, y)$  containing  $x$  and  $y$ .

### Closure with splitters

While there exists a splitter add it to the set.

### Complexity

$$O(n^2 \cdot (n + m)) = O(n^2 m)$$

This can be improved to  $O(nm)$  using usual tricks.

## Primality testing

One can derive a primality test since if there exists a non trivial module, it contains at least two vertices.

For some problems Bottom-Up techniques are the best known.

## Origins : Golumbic, Kaplan, Shamir 1995

**Input** :  $G_1 = (V, E_1)$  and  $G_2 = (V, E_2)$  2 undirected graphs such that  $E_1 \subseteq E_2$  and  $\Pi$  be a graph property.

**Results** : a sandwich graph  $G_s = (V, E_s)$  satisfying property  $\Pi$  and such that  $E_1 \subseteq E_s \subseteq E_2$ .

Edges of  $E_1$  are forced, those of  $E_2$  are optional ones, but those of  $E_3 = \overline{E_2}$  are forbidden.

Unfortunately most cases are NP-complete, as for example of  $\Pi$

- ▶  $G_s$  being comparability, chordal, strongly chordal, ...

## Only few polynomial cases

- ▶ cographs Golubic, Kaplan, Shamir (1995)
- ▶ sandwich module Cerioli, Everett, de Figueiredo, Klein (1998)
- ...

## Natural question

Find efficient algorithms for these polynomial cases.

## Sandwich module problem

**Input** :  $G_1 = (V, E_1)$  and  $G_2 = (V, E_2)$  2 undirected graphs such that  $E_1 \subseteq E_2$ .

**Result** : a sandwich graph  $G_s = (V, E_s)$  having a non trivial module and such that  $E_1 \subseteq E_s \subseteq E_2$ .

## Minimal Sandwich Module

### Splitter

For a subset  $A \subseteq V$ , a splitter is a vertex  $z \notin A$

s.t.  $\exists x, y \in A$  with  $zx \in E_1$  and  $zy \notin E_2$  (or equivalently  $zy \in E_3$ )

A splitter is also called **bias vertex**.

### Algorithm

The computation of a minimal sandwich module can be done in  $O(n^2 \cdot (n + m_1 + m_3))$ .

Hard to do better with this idea, using a bottom up approach.

## Brute Force Algorithm

Using the decomposition theorem, we only have to compute at most  $n$  times some connected components of  $G$  or its complement.  
 $O(n \cdot (n + m))$  complexity.

## Comparability graphs

Graph that admits a transitive orientation. Not all graphs are comparable graphs. The 3-sun and its complement are not comparable graphs.

Characterization theorem : Ghouila-Houry 1962, Gimore Hoffmann 1964

Every odd pseudo-cycle  $[a_1, a_2, \dots, a_{2q+1}, a_1]$  has a chord  $[a_i, a_{i+2}] \pmod{2q+1}$ .

$\mu = [a_1, a_2, \dots, a_{2q+1}, a_1]$  is a pseudo cycle if every consecutive vertices of  $\mu$  are joined by an edge of  $G$ .

### Proof

The necessary condition is obvious since following this induced odd pseudo-cycle we are forced to alternate the orientation of the edges, which is impossible since the number is odd. We will prove the sufficiency algorithmically.

## $\Gamma$ -classes

### Definition from Golumbic

We define a binary relation  $\Gamma$  on the edge set as follows :

For  $ab, ac \in E$ ,  $ab\Gamma ac$  if  $bc \notin E$ .

The equivalence classes of  $\Gamma^t$  are called the forcing classes of  $G$ .

Their orientation are forced in any transitive orientation of  $G$ .

For a complete graph, each edge is a forcing class.

### Relationships with modules

- ▶ If a forcing class  $F$  has an edge inside a module  $M$ , then  $F \subseteq G(M)$ .
- ▶ The set of vertices covered by a forcing class  $F$  denoted by  $V(F)$  is a module.

## Consequences

- ▶ If  $G$  is prime, for any forcing class  $F$ ,  $V(F) = V$ .
- ▶ NB the converse is false.
- ▶  $\forall M$  module of  $G$ ,  $M = \cup_{F|F \subseteq G(M)} V(F)$ .

## Transitive orientations

A comparability graph is a graph which admits a transitive orientation. Not all graphs are comparability graphs.

1. A prime comparability graph has a unique transitive orientation (up to reversal). Converse false.
2. The set of all transitive orientations of a comparability graph can be polynomially computed from its modular decomposition tree.

## Theorem

If  $G$  and  $\overline{G}$  are connected, it exists a unique forcing class  $F$  such that :  $V(F) = V$ .

## Proof

Let us first consider a prime graph  $G$  with more than 4 vertices.  $G$  contains a  $P_4$   $abcd$ . Necessarily  $ab, bc, cd$ , belongs to the same forcing class  $F$ . If  $V = \{a, b, c, d\}$  we are done. Else it exists another edge  $zt \notin \{a, b, c, d\}$ . Since  $G$  is connected let us consider a shortest path from  $zt$  to the  $P_4$ . Suppose  $zt \notin F$ . Using this shortest path either we conclude  $zt \in F$  or  $u$  the last vertex of the path is connected to  $\{a, b, c, d\}$  with edges that do not belong to  $F$ . Since  $V(F) = V$ , necessarily it exists a chain of implication from some edge of the  $P_4$  to some edge  $uw$  which yields a contradiction. So we have proven the uniqueness.

## End of the proof

Then if  $G$  is not prime, it admits a prime decomposition  $H$  which has the property above, since for every maximal module  $M$ ,  $G(M)$  and  $\overline{G(M)}$  are connected, it is easy to see using the  $P_4$  contained in  $H$  all the edges of  $G$  corresponding to edges of  $H$  belong to the same forcing class.

## Sketch of a modular decomposition algorithm

1. Compute the forcing classes of  $G$  and  $\overline{G}$ .
2. For each connected component  $G_i$  of  $G$ , select the unique forcing class  $F$ , such that  $V(F) = V_i$ .  
Compute the connected components of  $G_i - F$  they are strong modules of  $G_i$   
Apply the same process on  $\overline{G}$  to obtain the maximal modules of  $G$ .
3. Recurse on each maximal module
4. Step 1 is the bottleneck and requires  $O(n^3)$ .

## Golumbic's simple transitive orientation algorithm

**Data:** a graph  $G = (V, E)$

**Result:** an orientation of the edges of  $G$

**while**  $E$  *not empty* **do**

    Pick an edge  $ab \in E$  and compute its forcing class  $F_{ab}$

**if**  $G(F_{ab})$  *is not transitively orientable* **then**

        STOP print : "  $G$  is not a comparability graph"

**end**

$E \rightarrow E - F_{ab}$

**end**

To test whether  $G(F_{ab})$  is not transitively orientable, since this graph is made up with a unique forcing class, it suffices to choose an orientation of an edge and follows the  $\Gamma$  relation. If an edge need to be oriented in both directions, then it fails.

## Two different related algorithmic problems

1. Computing a transitive orientation (it exist a linear time algorithm).
2. Testing if an orientation is transitive (best known algorithms run in  $O(n.m)$  or use boolean matrix multiplication  $O(n^\alpha)$ ).

## Partition Lattice

### Definition

Let  $P = \{X_1, \dots, X_n\}$  and  $Q = \{Y_1, \dots, Y_m\}$  be two partitions of  $V$ .

$P \leq Q$  iff  $\forall j, 1 \leq j \leq m \ Y_j = \cup_{i \in I} X_i$  for some set  $I \subseteq \{1, n\}$

## Refining a partition

### Definition

Let  $S \subseteq V$ , and  $P = \{X_1, \dots, X_n\}$  be a partition of  $V$ .

$Q = \text{Refine}(S, P) = \{X_1 \cap S, X_1 - S, \dots, X_n \cap S, X_n - S\}$

$S$  is called a pivot.

*NB Some sets can be empty.*

- ▶  $\text{Refine}(S, P) \leq P$
- ▶  $\text{Refine}(S, P) = P$  iff  $S$  is an union of parts of  $P$

### Duality

$\text{Refine}(S, P) = \text{Refine}(\overline{S}, P)$

- └ Transitive orientations and modular decomposition
  - └ Partition refinement techniques

## Complexity

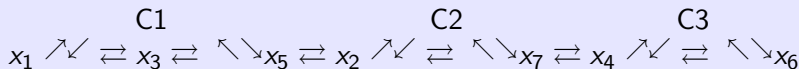
Can be computed in  $O(|S|)$ , using an ad hoc data structure.

# Implementation

$$V = \{x_1, \dots, x_7\}$$

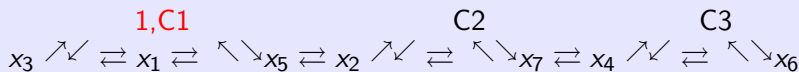
$$P = \{C1, C2, C3\} \text{ and } S = \{x_3, x_4, x_5\}$$

## Data Structure

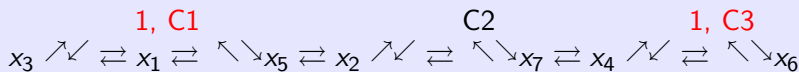


## First Step

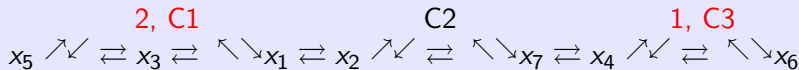
### Processing $x_3$



### Processing $x_4$



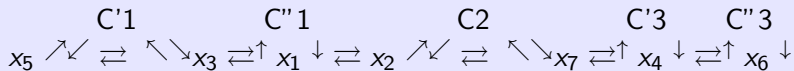
### $x_5$



## Second step

Maintain a list of the  $C_i$ 's that intersect  $S$ . List bounded by  $|S|$ .

### Result



$Refine(P, S) = \{C'1, C''1, C2, C'3, C''3\}$

computed in  $O(|S|)$

## Exercise

Find a linear-time **stable** implementation, i.e. preserving in each part some previous given ordering.

The is useful for implementing  $LexBFS^+$ .

## The particular case of twin vertices

### Algorithm

Start with  $P = \{V\}$

For each vertex  $x \in V$

**Do**  $P = \text{Refine}(N(x), P)$

### Proof

At the end of the algorithm no part of the partition can be splitted by a vertex.

Therefore the parts of  $P$  are made up with false twins.

### Complexity

$O(n + m)$

## Generic Refinement Algorithm

**Input** :  $P$  a partition and  $\mathcal{S}$  a set of pivots

**While**  $\mathcal{S} \neq \emptyset$

**Choose**  $S \in \mathcal{S}$

$P = \text{Refine}(S, P)$

add all new generated parts to  $\mathcal{S}$

This technique is very powerful not only for graph algorithms.

First used by Corneil for Isomorphism Algorithms 1970

Hopcroft Automaton 1971

Crochemore string sorting 1981

...

## Applications

- ▶ QUICKSORT
- ▶ Minimal deterministic automaton, Hopcroft  $O(n \log n)$  1971.
- ▶ Relational coarset partition, Paige, Tarjan 1987
- ▶ Doubly Lexicographic ordering of a boolean matrix, Paige, Tarjan 1987  $O(L \log L)$ .  
using a 2-dimensional refinement technique.
- ▶ Interval graph recognition, modular decomposition, many problems on graphs (LexBFS ...). 1990 –

## Vertex splitting

Also called vertex partitioning

Use  $N(x)$  as a pivot set.

## A linear algorithm ?

It exists simple algorithms in  $O(n + m \log n)$  in which the neighbourhood of a given vertex can be used to refine the partition at most  $\log n$  times.

How to improve this to a constant number ?

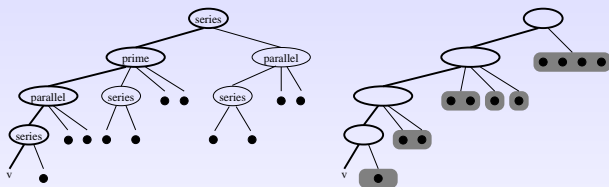
- └ Transitive orientations and modular decomposition
- └ Partition refinement techniques

## Three explored directions

- ▶ Ehrenfeucht et al approach
- ▶ Using Factoring Permutation
- ▶ Using LexBFS (as for cographs)

## Ehrenfeucht et al approach

$\mathcal{M}(G, v)$  is composed by  $\{v\}$  and the maximal modules of  $G$  not containing  $v$ .



### Principle of the Ehrenfeucht et al.'s algorithm

1. Compute  $\mathcal{M}(G, v)$
2. Compute  $MD(G/\mathcal{M}(G, v))$
3. For each  $\mathcal{X} \in \mathcal{M}(G, v)$  compute  $MD(G[\mathcal{X}])$

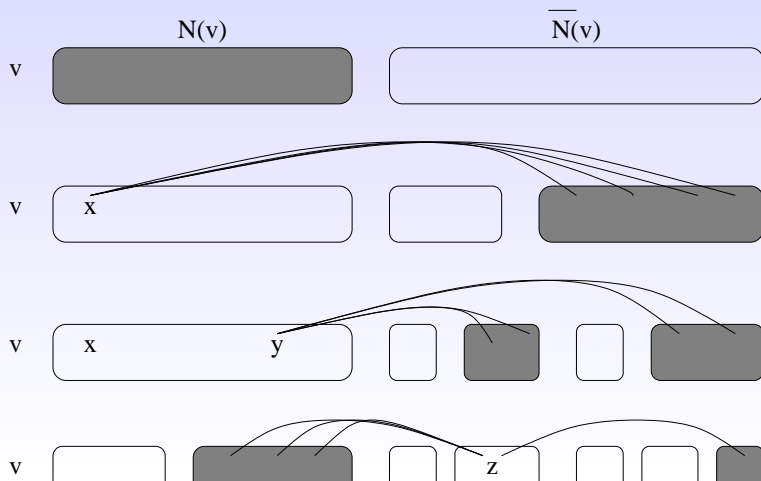
## Computing $\mathcal{M}(G, v)$ via Partition Refinement

### Splitter again

If  $z$  is a splitter of  $A \subseteq V$  then any strong module contained in  $A$  is either contained in  $N(z) \cap A$  or in  $A - N(z)$ .

Computation of  $\mathcal{M}(G, v)$ 

$\Rightarrow O(n + m \log n)$  time using vertex partitioning algorithm.

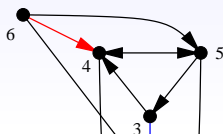
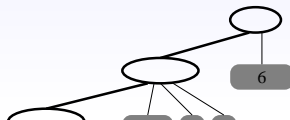
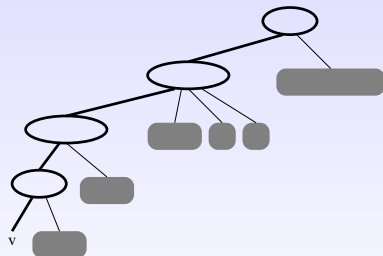


How to reconstruct the modular decomposition tree from the partition  $\mathcal{M}(G, \nu)$ ?

The most difficult step in many algorithms.

## Computation of $MD(G/\mathcal{M}(G,v))$

- ▶ The modules of  $G/\mathcal{M}(G,v)$  are linearly nested :  
any non-trivial module contains  $v$
- ▶ The *forcing graph*  $\mathcal{F}(G, v)$  has edge  $\vec{xy}$  iff  $y$  separates  $x$  and  $v$



- └ Transitive orientations and modular decomposition
  - └ Partition refinement techniques

## Complexity

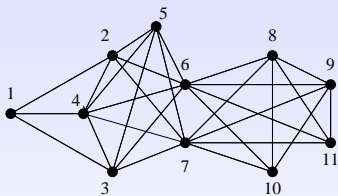
- ▶ [Ehrenfeucht et al.'94] gives a  $O(n^2)$  complexity.
- ▶ [MS00] gives a very simple  $O(n + m \log n)$  algorithm based on vertex partitioning.
- ▶ [DGM'01] proposes a  $O(n + m \cdot \alpha(n, m))$  and a more complicated  $O(n + m)$  implementation.

## Other algorithms

- ▶ [CH94] and [MS94] present the first linear algorithms.
- ▶ [MS99] present a new linear time algorithm which extends to transitive orientation.

## Factoring Permutation

A **factoring permutation** of a graph  $G = (V, E)$  is a permutation of  $V$  in which any strong module of  $G$  is a factor. [CH 97]



1 2 3 4 5 6 7 8 9 10 11



## Splitter interpretation

### An $O(m \log n)$ algorithm

Starting with the partition  $\{N(x), \{x\}, \overline{N(x)}\}$ , we maintain the following invariant :

It exists a factoring permutation smaller than the current partition.

### The smallest half Hopcroft's rule

When a part is divided, select only the smallest half as pivot.

This yields that for every vertex  $x$ ,  $N(x)$  is visited at most  $\log n$  times.

The algorithm is correct since if every part of a partition has been used as pivot, the last one is necessarily a module.

## Transitive orientation in $O(m \log n)$

Let  $G$  be comparability graph

Starting with the partition  $\{\{x\}, \overline{N(x)}, N(x)\}$ ,  
in which  $x$  is the last vertex on a LexBFS on  $\overline{G}$

we maintain the following invariant :

It exists a linear extension of  $P(G)$  smaller than the current partition.

### The smallest half Hopcroft's rule

When a part is divided, select only the smallest half as pivot.

This yields that for every vertex  $x$ ,  $N(x)$  is visited at most  $\log n$  times.

The algorithm is correct since if every part of a partition has been used as pivot, the last one is necessarily a module.

- └ Transitive orientations and modular decomposition
- └ Partition refinement techniques

### LexBFS on co-comparability graphs

If  $G$  is a comparability graph and  $x$  the last vertex of a LexBFS applied on  $\overline{G}$ , then  $x$  can be taken as a source or a sink in some transitive orientation of  $G$ .

### LexBFS and modules

In a LexBFS when the tie-break set is the last part of the partition, this set is necessarily a module.

The two previous algorithms are nearly similar, only changes the insertion rules during the refinement !!

But we need an implementation of partition refinement with a notion of left and right in the partition, and a notion of smallest half.

# Cographs

## Recursive Definition

The class of cographs is the smallest class of graphs containing  $G_0$  and closed under series and parallel compositions.

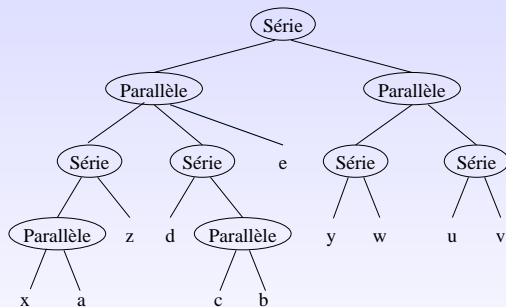
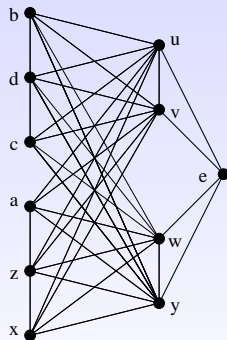
## Characterisation Theorem

A graph is a cograph iff it does not contain a  $P_4$ .

## Obvious

With the structure of prime graphs.

## An example



# Cotree

## Definition

The modular decomposition tree of a cograph is called a **cotree**.

## Properties of the cotree

- ▶ Vertices of the cotree can be labelled with 0 (parallel) and 1 (series).
- ▶ From  $G$  to  $\overline{G}$  just exchange 0's and 1's.
- ▶  $xy \in E$  iff  $LCA(x, y)$  in the cotree is labelled with 1

## Un schma simplicial pour les cographes

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



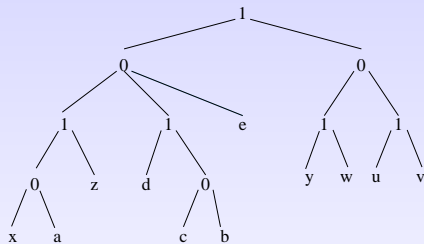
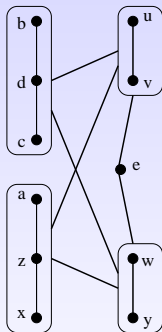
Expliquer le coup d'un raffinement par sommet ....

1. Best known complexity :  $O(n + m \log n)$  and  $O(n + m)$  for cographs
2. Recent algorithmic results around this notion [UY00] [BXHP05] [BCdMR05]
3. For some graph families, computing  $MD(G)$  from a factoring permutation costs  $O(n)$  time

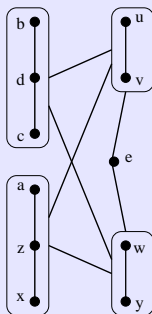
## The best Algorithm Using LexBFS

A cograph recognition algorithm [BCHP03]

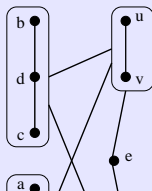
1.  $\sigma \leftarrow \text{LexBFS}(G)$
2.  $\bar{\sigma} \leftarrow \text{LexBFS}^-(\bar{G}, \sigma)$
3. **If**  $\sigma$  and  $\bar{\sigma}$  both have the NS-property **then**
  - 3.1 Answer "  $G$  is a cograph"
  - 3.2 Build  $MD(G)$
4. **Else** Output a  $P_4$



## Computing a LexBFS ordering $\sigma$



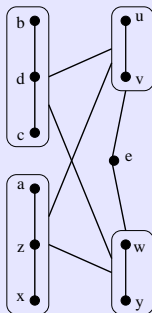
x y w z u v a d c b e



x y w z u v a d c b e

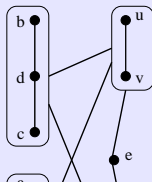
x y w z u v a d c b e

## Computing $\bar{\sigma} = \text{LexBFS}^-(\overline{G}, \sigma)$



x y w z u v a d c b e

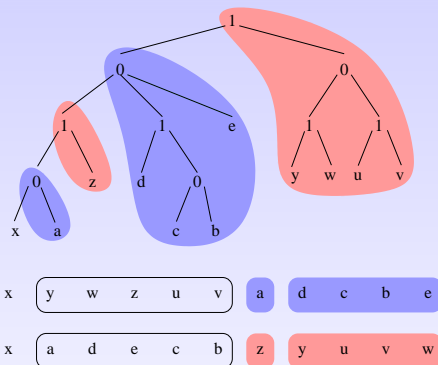
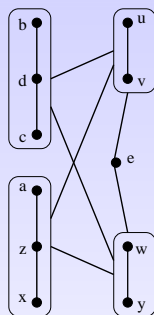
x a d c b e y w z u v



x y w z u v a d c b e

x a d c b e y w z u v

x a d c b e y w z u v



## Lemma

$\mathcal{M}(G, x)$  is composed by the slices  $S_i(x)$  of  $\sigma$  and  $\overline{S}_j(x)$  of  $\overline{\sigma}$ .

## Theorem

If  $G$  is a cograph, then  $MD(G)$  can be retrieved from the slice structure of  $\sigma$  and  $\overline{\sigma}$ .

## Generalization to arbitrary graphs ?

1. There are many similarities between two-LexBFS-sweep algorithm and the linear implementation of Ehrenfeucht et al.'s algorithm [DGM01]. **We conjecture that there is way to obtain the modular decomposition tree using only graph searching.**
2. LexBFS is useful for the transitive orientation problem. Could it lead to a simple linear time algorithm for this problem ?