Recognition Algorithms
MPRI 2017–2018

Michel Habib
habib@irif.fr
http://www.irif.fr/~habib

Sophie Germain, septembre 2017

Schedule

Algorithmic aspects of modular decomposition

- Historical Notes
- Bottom up Techniques
- Top Down techniques

Interval recognition

Recognition of permutation graphs

Applications of modular decomposition

- A very natural operation to define on discrete structures, searching for regularities.
- A structure theory for comparability graphs
- 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.

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 $O(logn), O(nm)$ proc. parallel, cographs, EREW-PRAM
- Spinrad 1992 $O(n + malpha(m, n))$
- Cournier, Habib 1993 $O(n + malpha(m, n))$
- 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, Sprinrad 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 $O(n+mlogn)$ via a factoring permutation
- McConnell, Spinrad 2000 $O(n+mlogn)$
- 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

- 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.

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 basic 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

$O(n^2.(n + m))$

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 Π be a graph property.
Results : a sandwich graph $G_s = (V, E_s)$ satisfying property Π 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 Π

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

Only few polynomial cases

- cographs Golumbic, 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 $O(n^2.(n + m_1 + m_3))$.

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. $O(n.(n+m))$ complexity.

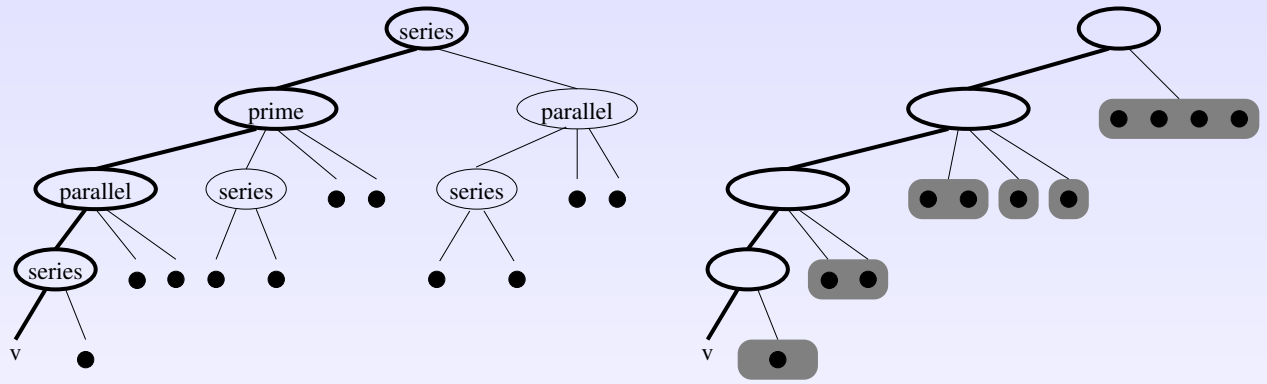
Three explored directions

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

Ehrenfeucht et al approach

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

$\mathcal{M}(G, v)$ is the partition of $V(G)$ composed by $\{v\}$ and the maximal modules of G that do not contain 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 part $\mathcal{X} \in \mathcal{M}(G, v)$ compute $MD(G[\mathcal{X}])$

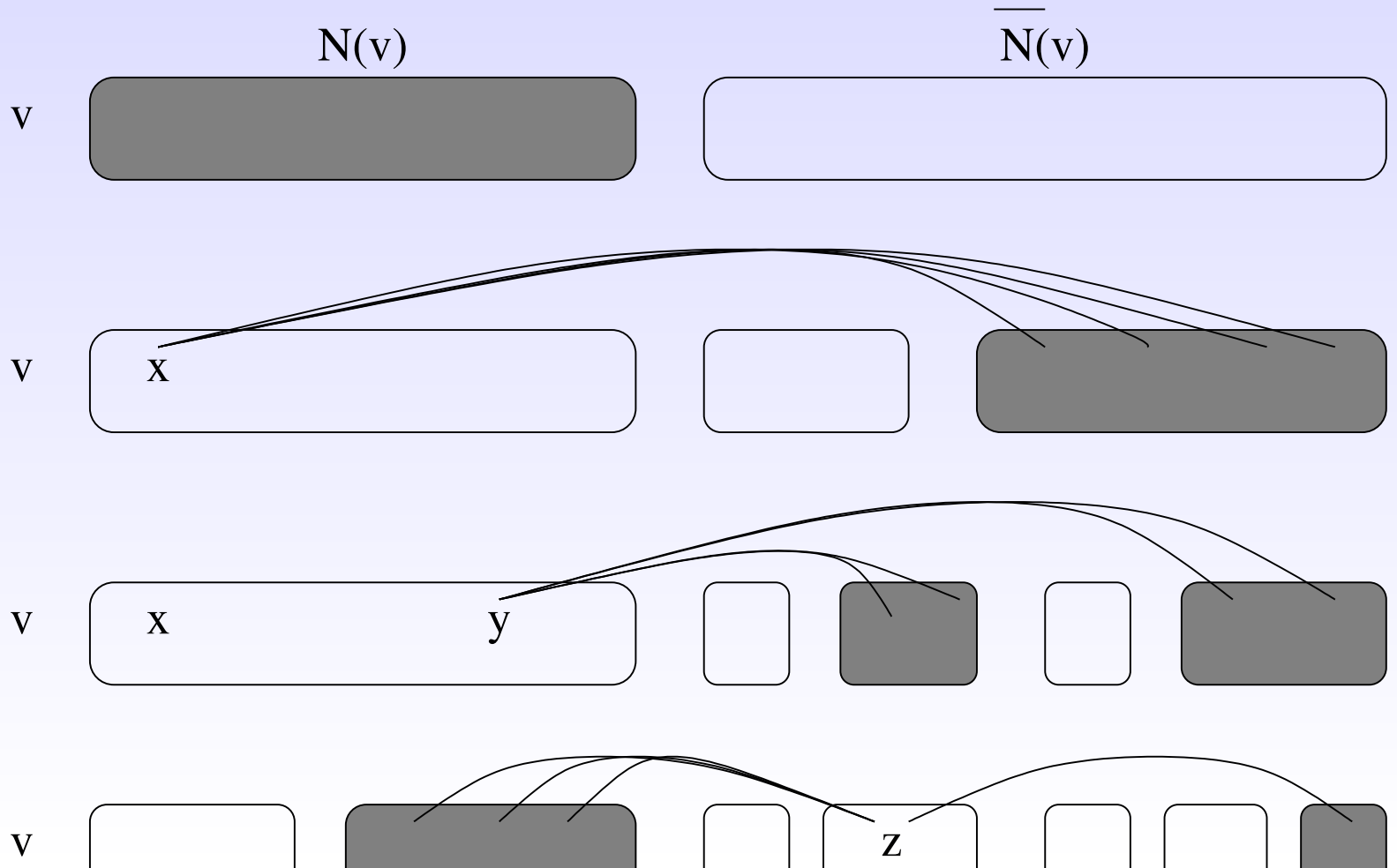
Computing $\mathcal{M}(G, v)$ via Partition Refinement

Splitter again

If z is a splitter of $A \subseteq V(G)$ 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.

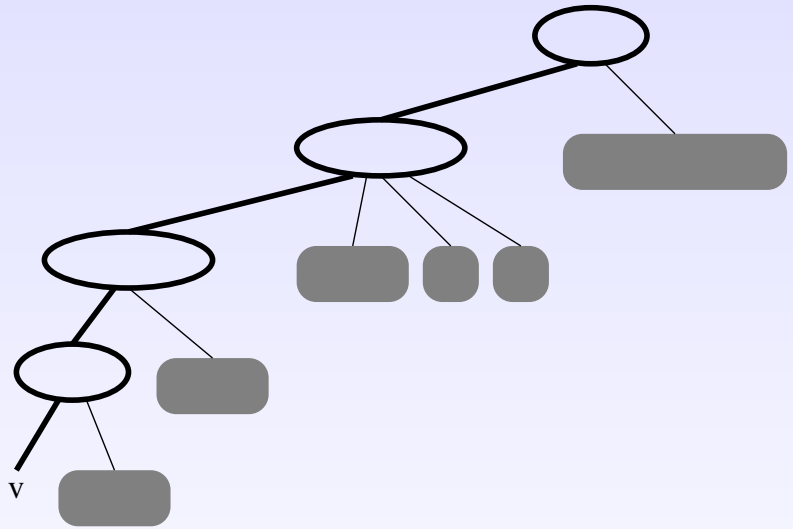


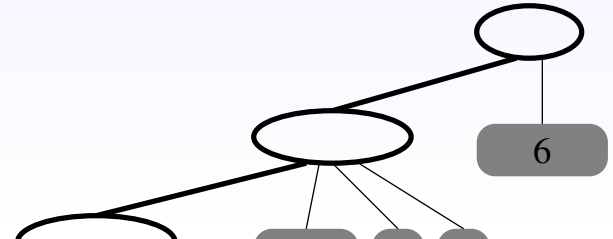
1. Particular partition refinement rule :
   Do not refine its part
   Just to maintain the invariant :
   Modular partition $\leq$ Current partition
2. To obtain a *logn*
   Avoid the biggest part

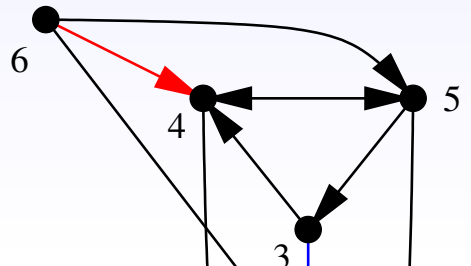
How to reconstruct the modular decomposition tree from the partition $\mathcal{M}(G, v)$?
The most difficult step in many algorithms.

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 $\overrightarrow{xy}$ iff y separates x and v







- The strong connected components of the *forcing graph* $\mathcal{F}(G, v)$ provides the modules of of $G_{/\mathcal{M}(G,v)}$.
- Recurse on each module.

Complexity

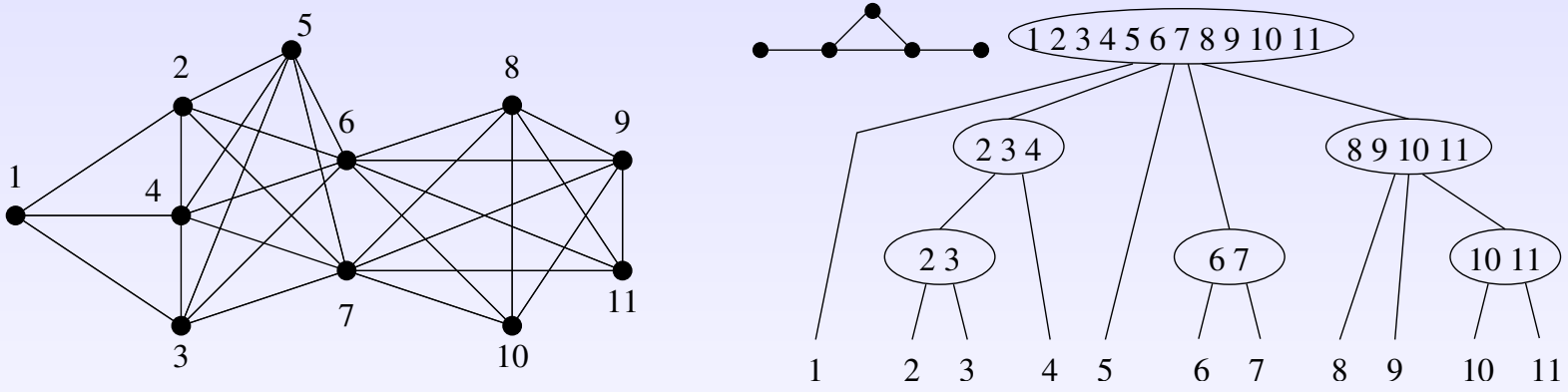
- [Ehrenfeucht et al.'94] gives a $O(n^2)$ complexity. It is quite tricky to efficiently compute the *forcing graph* $\mathcal{F}(G, v)$.
- [MS00] gives a very simple $O(n + m \log n)$ algorithm based on vertex partitioning.
- [DGM'01] proposes a $O(n + m.\alpha(n, m))$ and a more complicated $O(n + m)$ implementation.

Other algorithms

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

Factoring permutations

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



A factoring permutation is simply a left-right ordering of the leaves of a plane drawing of $MD(G)$.

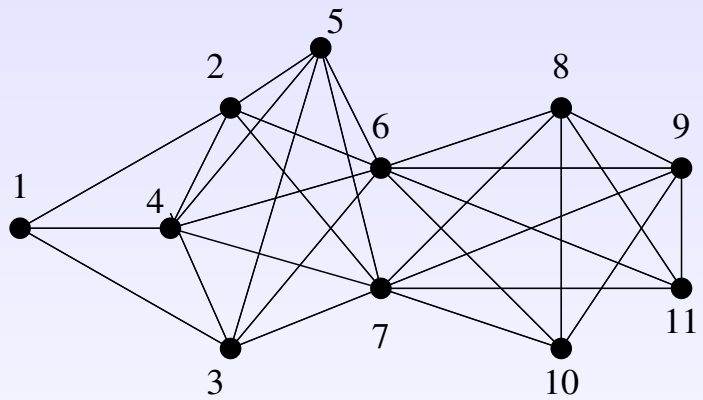
Consequence : it always exists factoring permutations.
There are easier to compute that the modular decomposition tree.

Invariant

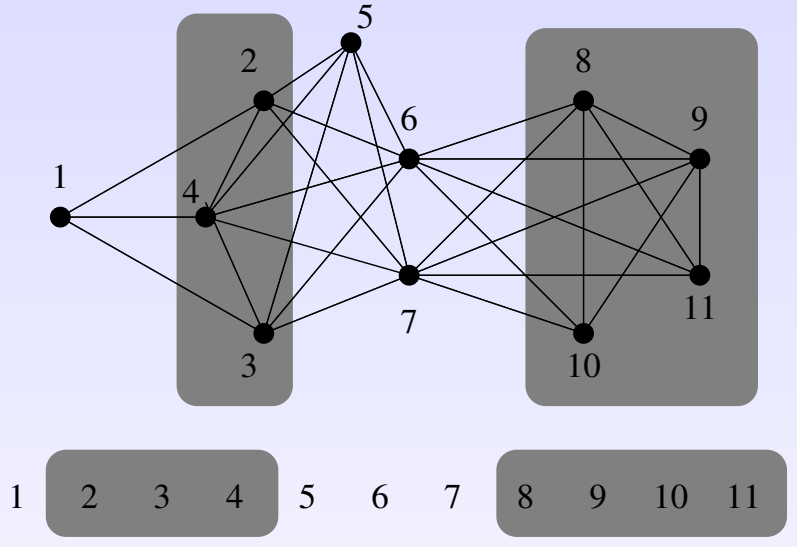
Any strong module is a factor of the partition.

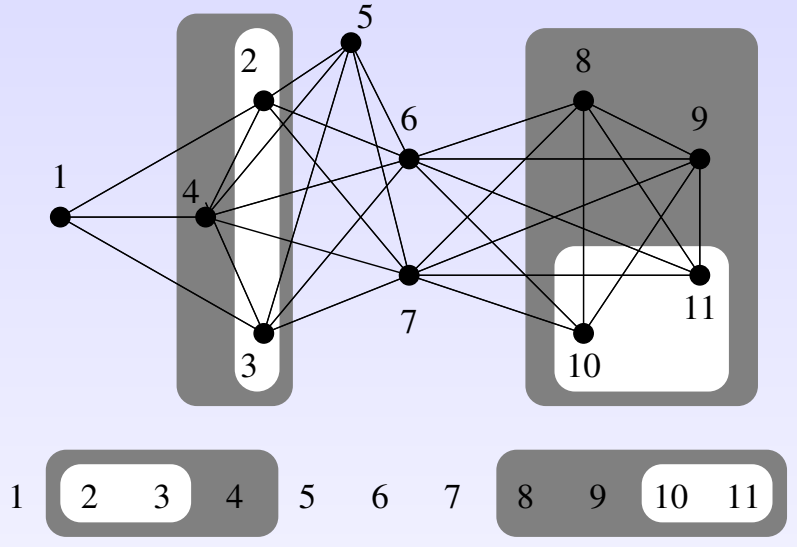
Factoring permutation

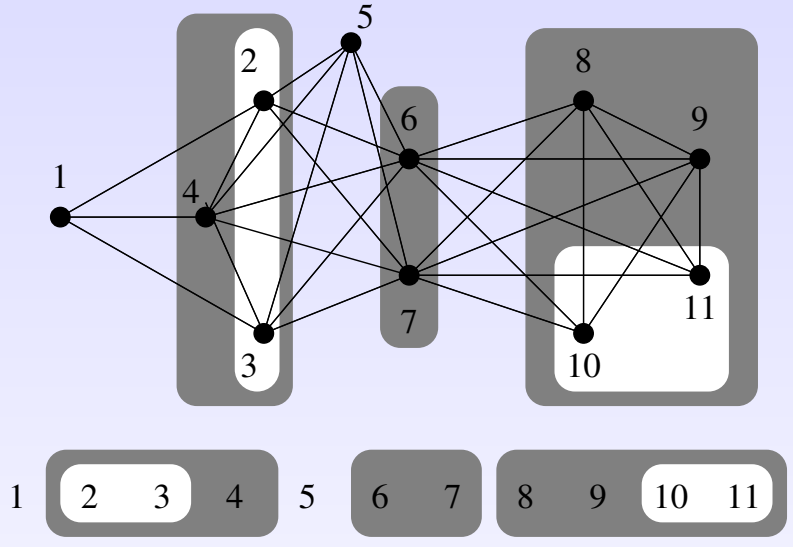
A **factoring permutation** of a graph $G = (V, E)$ is a permutation of V in which any strong module of G is a factor. [CH 97]



1 2 3 4 5 6 7 8 9 10 11


1
2
3
4
5
6
7
8
9
10
11
1 2 3 4 5 6 7 8 9 10 11


1
2
3
4
5
6
7
8
9
10
11
1 2 3 4 5 6 7 8 9 10 11


1
2
3
4
5
6
7
8
9
10
11
1 2 3 4 5 6 7 8 9 10 11

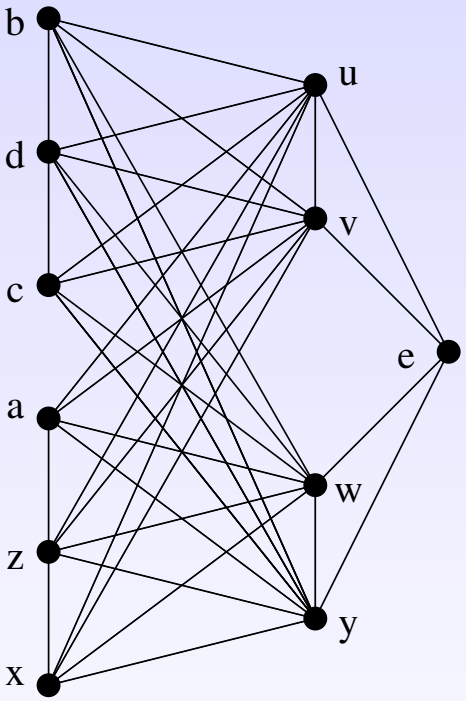
- From G to factoring permutation : $O(n + m \log n)$ [HPV99]
- From factoring permutation to $MD(G)$: $O(n + m)$ [CdMH01] [UY00] [BXHP05]

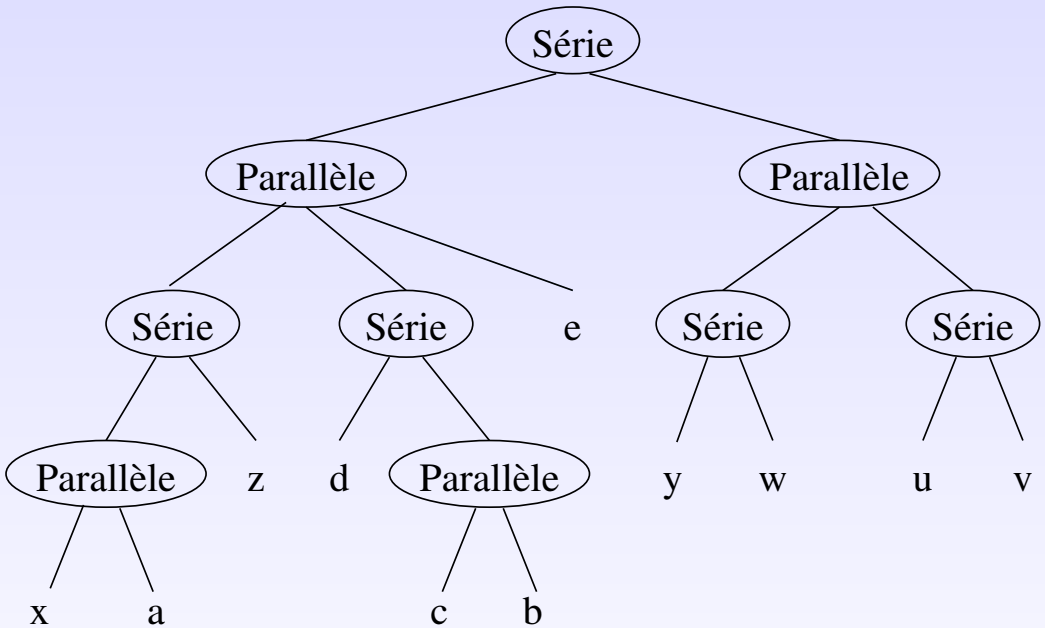
Factoring permutation

Factoring permutation (of cographs) via vertex partitioning

Starting with the partition $\{\overline{N(x)}, \{x\}, N(x)\}$, we maintain the following invariant :
It exists a factoring permutation smaller than the current partition.

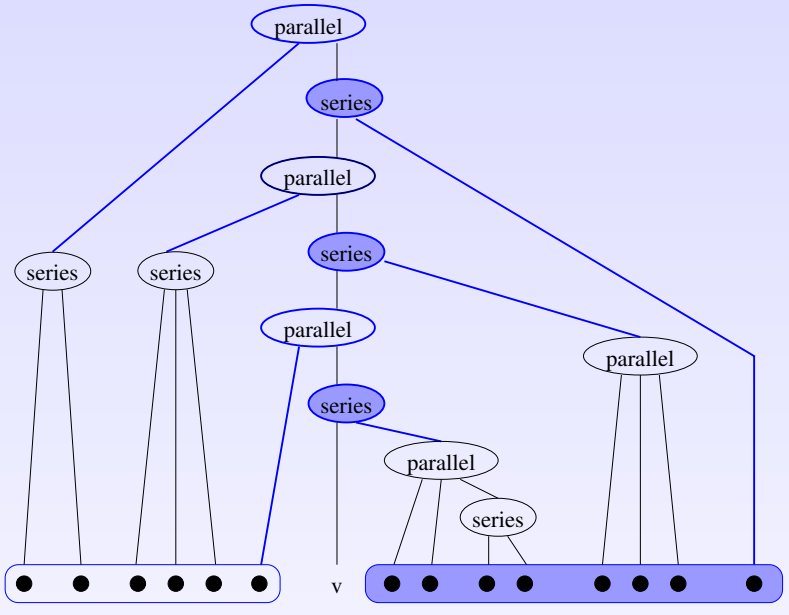
Recall of the tree structure of cographs

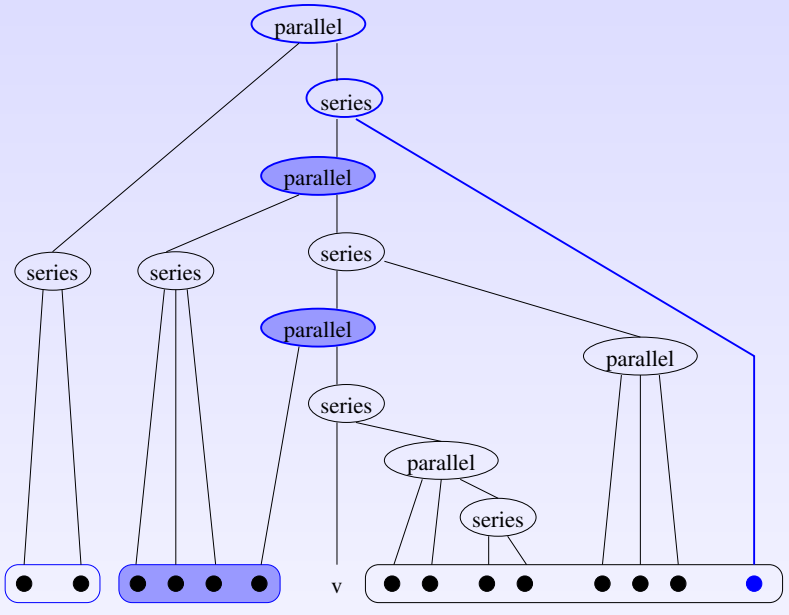


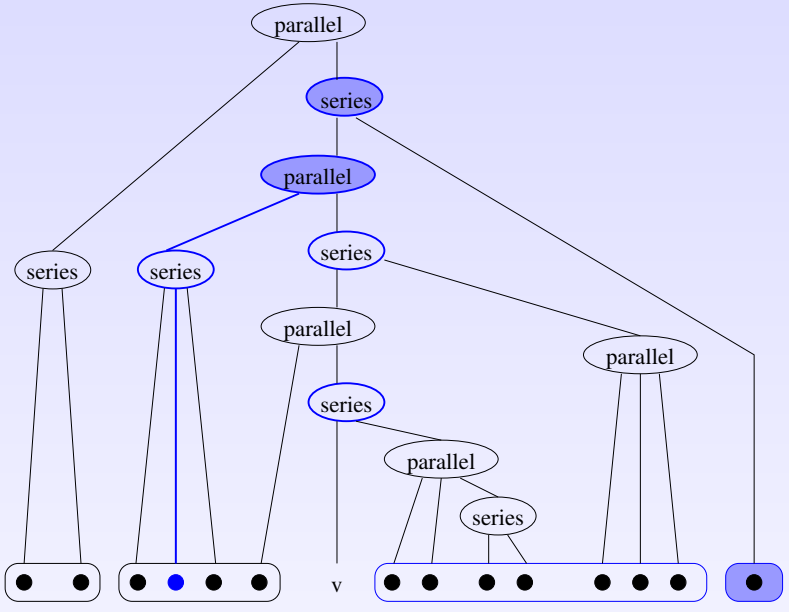


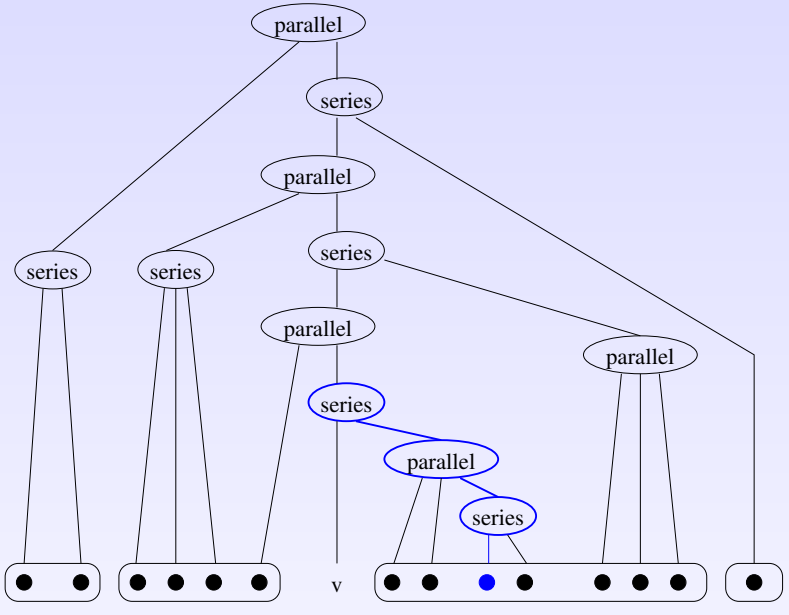
Splitter interpretation

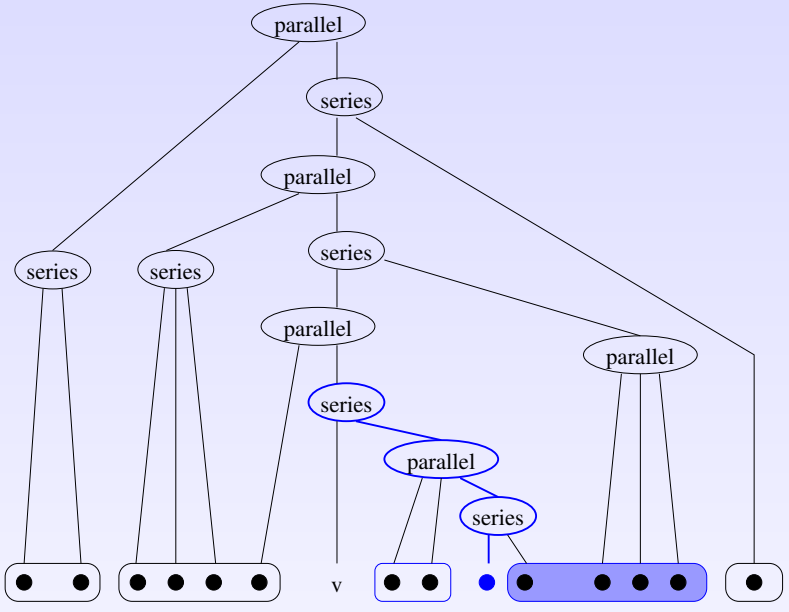
Starting with the partition $\{N(x), \{x\}, \overline{N(x)}\}$, we maintain the following invariant :
There exists a factoring permutation smaller than the current partition.


parallel
series
parallel
series
series
series
parallel
parallel
series
parallel
series
v


parallel
series
parallel
series
series
series
parallel
parallel
series
parallel
series
v


parallel
series
parallel
series
series
series
parallel
parallel
series
parallel
series
v


parallel
series
parallel
series
series
series
parallel
parallel
series
parallel
series
v


parallel
series
parallel
series
series
series
parallel
parallel
series
parallel
series
v

Invariant

Any strong module is a factor of the partition.

1. Brute force, just partition refinement. Using recursivity. $O(nm)$
2. Using the rule : take the smallest half $O(mlogn)$
3. Using at most one vertex per part to refine the other parts.
   Idea : only two parts (the extreme ones) refine nothing. Then we can restart the procedure to the closest one to the initial pivot.
   In the whole a vertex is at most used twice as a pivot.
   $O(n + m)$

Modular decomposition algorithms via partition refinement are very similar than the cograph recognition algorithms just a little more complicated.

[]


Anna Bretscher, Derek G. Corneil, Michel Habib, and Christophe Paul.
A simple linear time lexbfs cograph recognition algorithm.
SIAM J. Discrete Math., 22(4) :1277–1296, 2008.


Michel Habib and Christophe Paul.
A simple linear time algorithm for cograph recognition.
Discrete Applied Mathematics, 145(2) :183–197, 2005.


Michel Habib and Christophe Paul.
A survey of the algorithmic aspects of modular decomposition.
Computer Science Review, 4(1) :41–59, 2010.

Characterization Theorem for interval graphs (Folklore)

(i) $G = (V, E)$ is an interval graph, i.e. ; G is the intersection graph of a family of intervals of the real line

(ii) There exists a total ordering τ of the vertices of V s.t. $\forall x, y, z \in G$ with $x \leq_\tau y \leq_\tau z$ and $xz \in E$ then $xy \in E$.

(iii) G admits a maximal clique tree which is a chain.

(iii) implies (i) is obvious, since the chain of maximal cliques gives the interval representation.
(i) implies (iii). From the interval representation it is easy $O(n)$ to compute the sequence of maximal cliques.
(i) implies (ii) : one can associate to every vertex x of G an interval $[\mathit{left}(x), \mathit{right}(x)]$ of the real line.
Let us define τ as the left sides ordering : $x \leq_\tau y$ iff $\mathit{left}(x) \leq \mathit{left}(y)$. It is easy to verify that τ satisfies the condition.
(ii) implies (i) : starting with τ, for every vertex x we associate an interval in τ $[x, \mathit{last}(x)]$, where $\mathit{last}(x)$ is its rightmost neighbor in τ. This provides an interval representation of G.

To recognize an interval graph, we just have to compute a maximal clique tree and check if it is a chain ?

Difficulty : an interval graph has many clique trees and among them some are chains

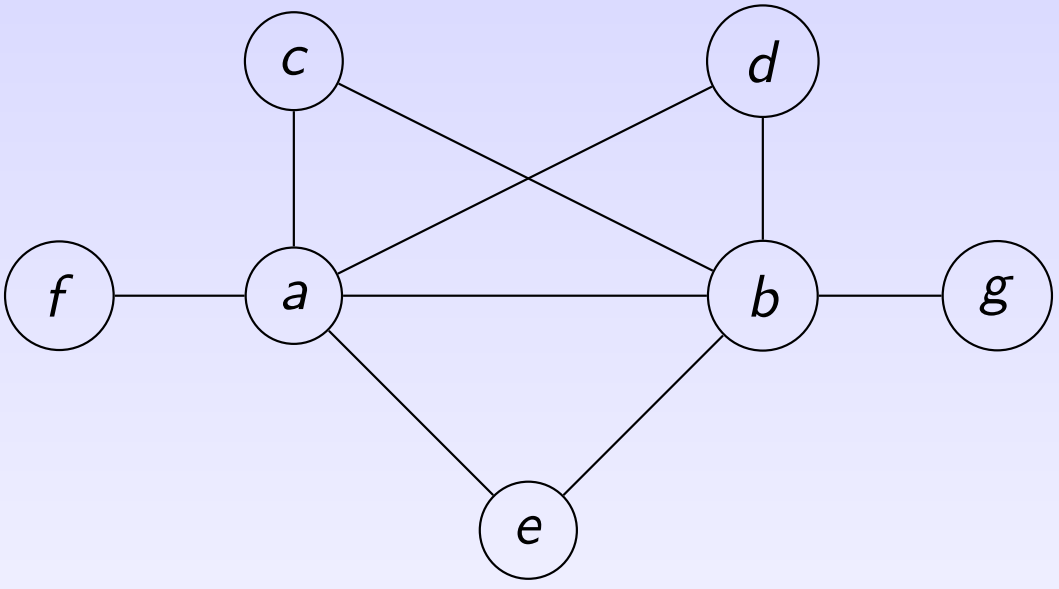


FIGURE: G a chordal graph

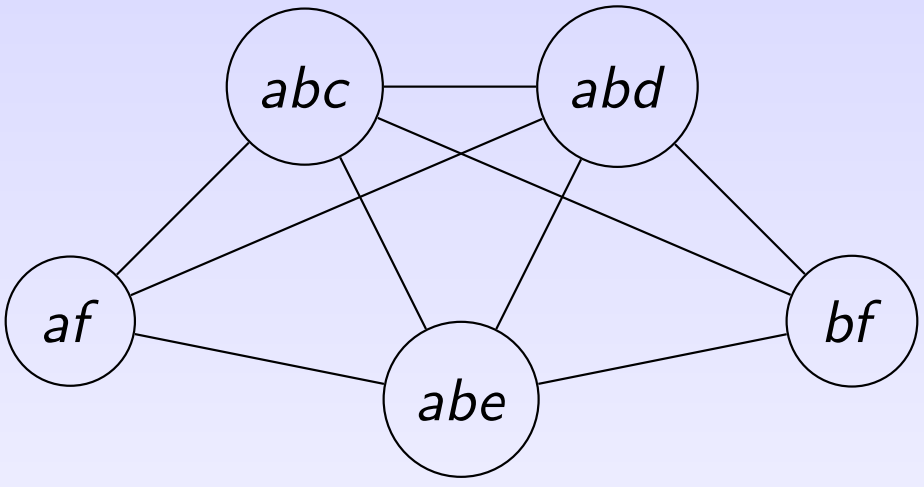


FIGURE: $\mathcal{C}_r(G)$ its reduced clique graph

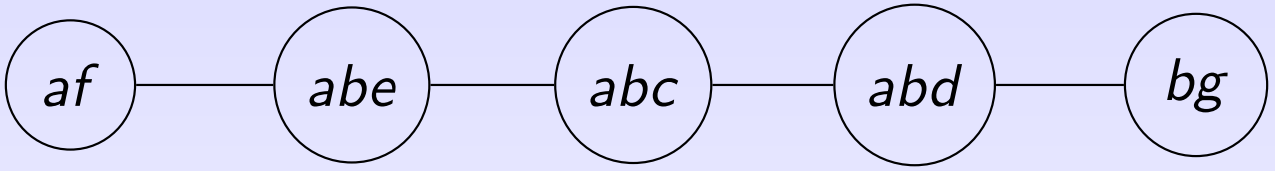


FIGURE: A good maximal clique tree T_1 showing that G is an interval graph

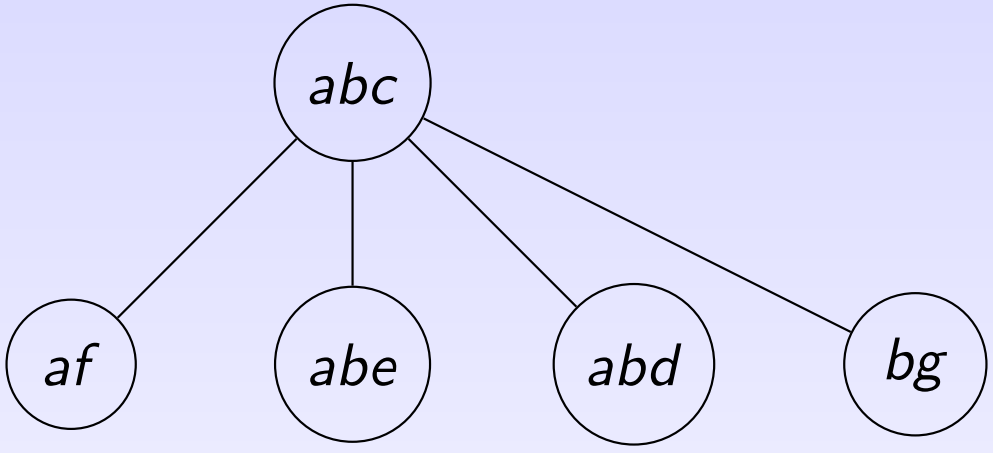


FIGURE: Another maximal clique tree T_2 obtained via LBFS $\sigma = c, a, b|d|e|f|g$

How can we transform T_2 to obtain T_1 ?

Many linear time algorithms already proposed for interval graph recognition ....
using nice algorithmic tools :
graph searches, modular decomposition, partition refinement, PQ-trees ...

Linear time recognition algorithms for interval graphs

- Booth and Lueker 1976, using PQ-trees.
- Korte and Mohring 1981 using LBFS and Modified PQ-trees.
- Hsu and Ma 1995, using modular decomposition and a variation on Maximal Cardinality Search.
- Corneil, Olariu and Stewart SODA 1998, using a series of 6 consecutive LBFS, published in 2010.
- M.H, McConnell, Paul and Viennot 2000, using LBFS and partition refinement on maximal cliques.
- P. Li, Y. Wu 2014, using a series of 4 kind of LBFS
- ...

A partition refinement algorithm working on maximal cliques

1. Compute a tree T using LBFS
   If T is not a maximal clique tree ; then G is not chordal, neither interval.
2. Start from the last maximal clique visited by the search
   Refine the cliques with the minimal separator.
3. Refine until each part is a singleton
4. If a part is not a singleton start recursively from the last clique of this part according to LBFS.
5. Check if the last partition is a chain of maximal cliques.

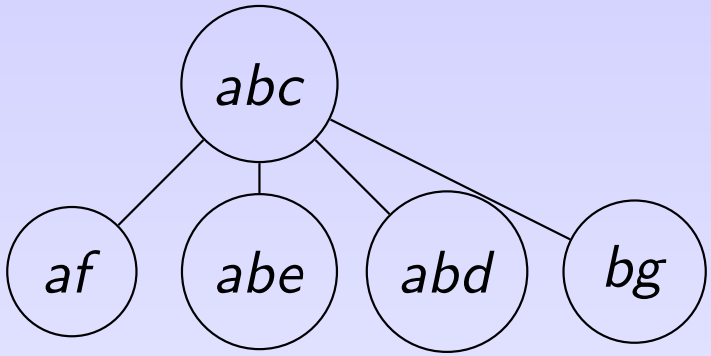


FIGURE: T_2 obtained via LBFS $\sigma = c, a, b|d|e|f|g$

- $\{bg\}|\{abc, abd, abe, af\}$
- refine with b gives :
  $\{bg\}|\{abc, abd, abe\}, |\{af\}$
- refine with a does not change the partition
- abe is the last maximal clique of the central part.
  $\{bg\}|\{abe\}|\{abc, abd\}, |\{af\}$
  refine with a, b does not change the partition
- abd is the last maximal clique of the central part.
  $\{bg\}|\{abe\}|\{abd\}|\{abc\}, |\{af\}$

Similar problems

1. Recognition of permutation graphs
2. Consecutive one property
3. Transitive orientation of a comparability graph
4. Planarity testing
5. Decomposition of boolean matrices
6. Robinsonian matrices
7. Other problems on symmetric positive matrices.

Recognition of Permutation graphs

PERMUTATION(G) :

Input: A connected graph $G = (V, E)$, cocomp ordering σ of G and cocomp ordering τ of $\overline{G}$

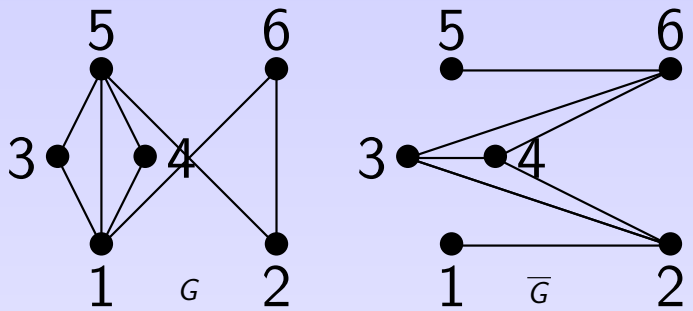
Output: The message that at least one of σ, τ is not a cocomp ordering of its graph $(G, \overline{G})$ or total orderings π_1, π_2^{dual} of $1, 2, \cdots |V|$ that certify that G is a permutation graph

$P_{G(\tau)} \leftarrow$ an acyclic orientation of G using τ ;

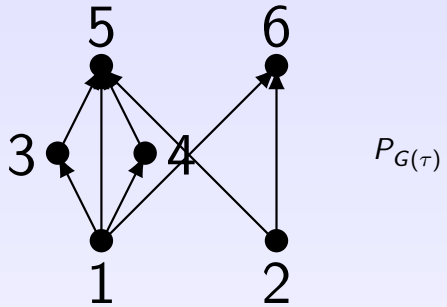
$\pi_1 \leftarrow dfgreedy^+(P_{G(\tau)}, \sigma)$;

$\pi_2 \leftarrow dfgreedy^+(P_{G(\tau)}, \sigma^{dual})$;

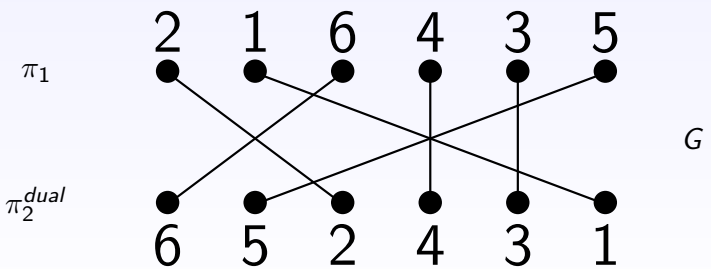
Check if π_1, π_2^{dual} represent G as a permutation graph ;



$\sigma = 1, 3, 5, 4, 2, 6$ is a cocomp ordering of G, but not a cocomp ordering of $\overline{G}$ (see umbrella (1, 5, 2))
$\tau = 1, 2, 3, 4, 5, 6$ is a cocomp ordering of $\overline{G}$, but not a cocomp ordering of G (see umbrella (2, 3, 6))



$\pi_1 = 2, 1, 6, 4, 3, 5 = \mathit{dfgreedy}(P_{G(\tau)}, \sigma)$
$\pi_2 = 1, 3, 4, 2, 5, 6 = \mathit{dfgreedy}(P_{G(\tau)}, \sigma^{\mathit{dual}})$



π_1 and π_2 are cocomp orderings of both G and $\overline{G}$

- dfgreedy is a graph search applied on posets and if σ is a cocomp then $\mathit{dfgreedy}^+(P_{G(\tau)}, \sigma)$ is also a cocomp.
- If G is a permutation graph, $\mathit{dfgreedy}^+(P_{G(\tau)}, \sigma)$ and $\mathit{dfgreedy}^+(P_{G(\tau)}, \sigma^{\mathit{dual}})$ provide a representation of G as a permutation graph.