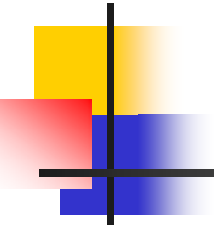
On co-distance hereditary graphs

Swan, Dubois, Vassilis Giakoumakis,

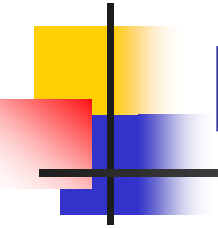
C.B. Ould El Mounir

MIS, Université d'Amiens, France



Plan

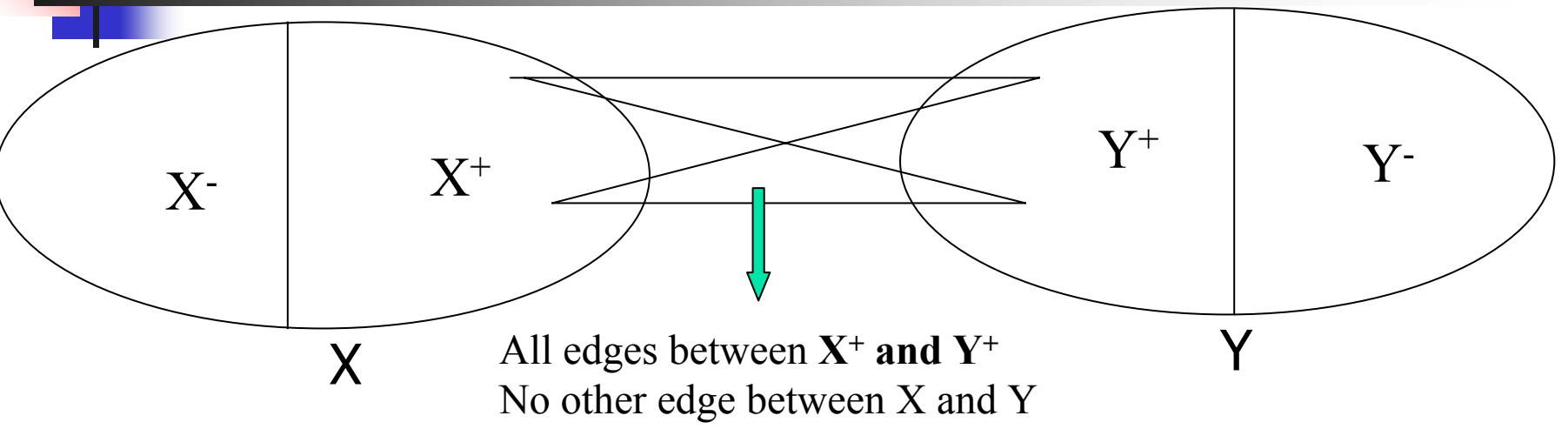
- Split decomposition
- Distance hereditary graphs : totaly split-decomposables
- Linear recognition of a co-distance hereditary graph
- Clique-width and k-expression of a co-distance hereditary graph



Notations

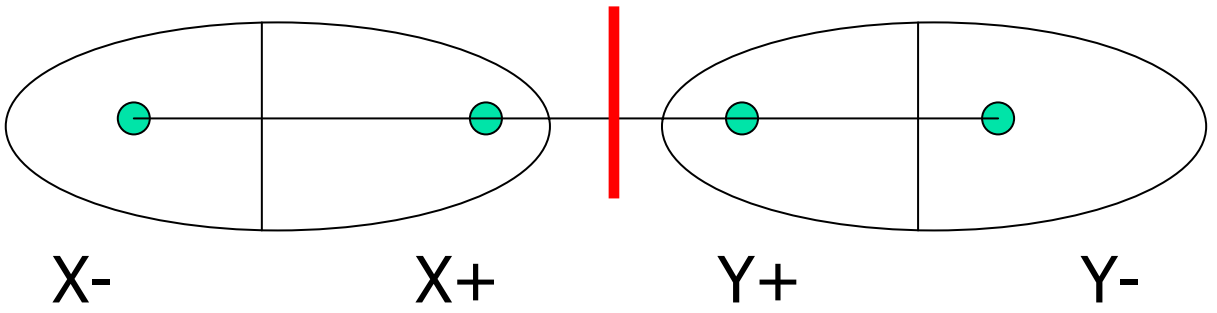
- All considered graphs are undirected without loop nor multiple edge.
- G=(V,E), V is the vertex set of G and E is its edge set G.
- co-G denote the complementary graph of G

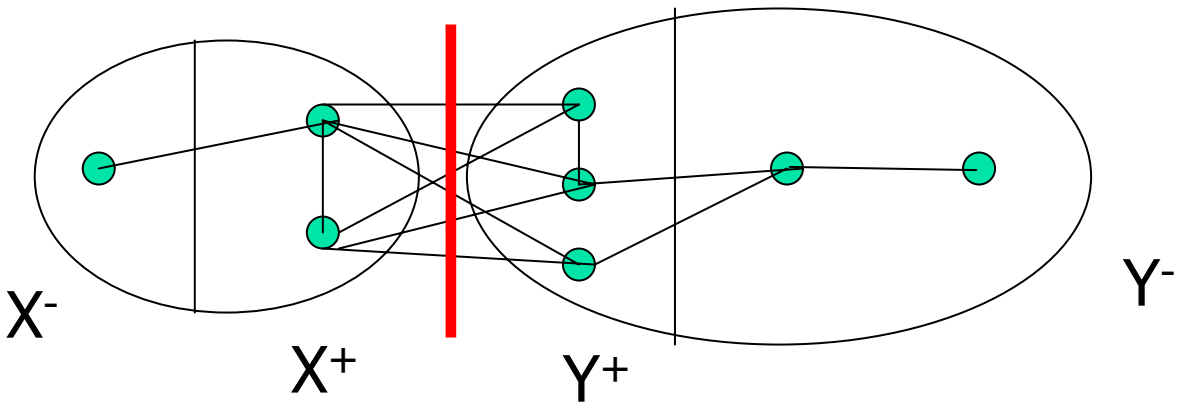
Split of a graph G



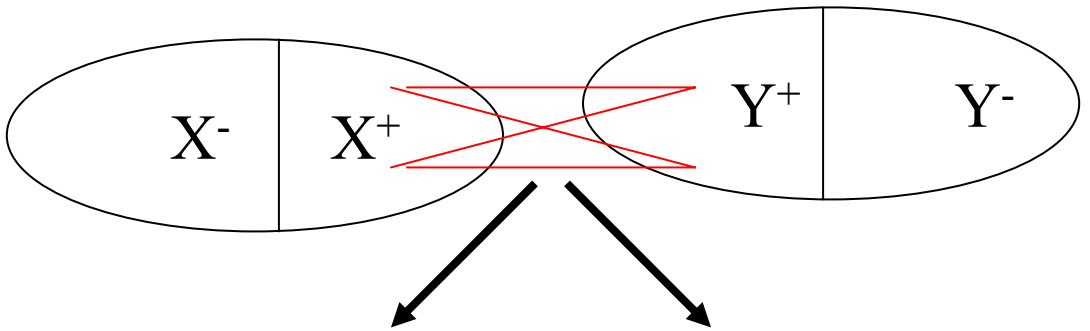
Each one of X and Y has at least two vertices

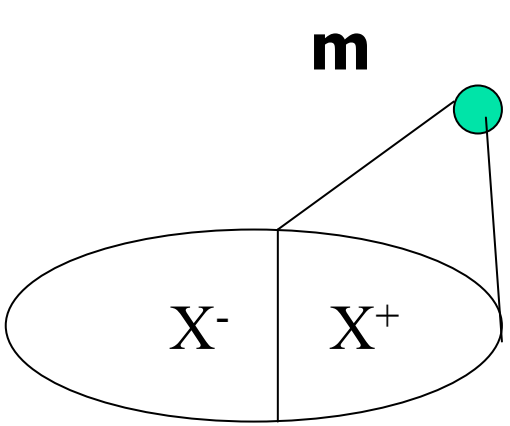
Examples



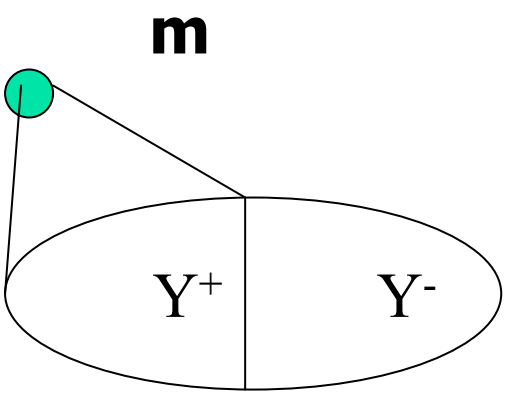


Decomposition-composition following splits



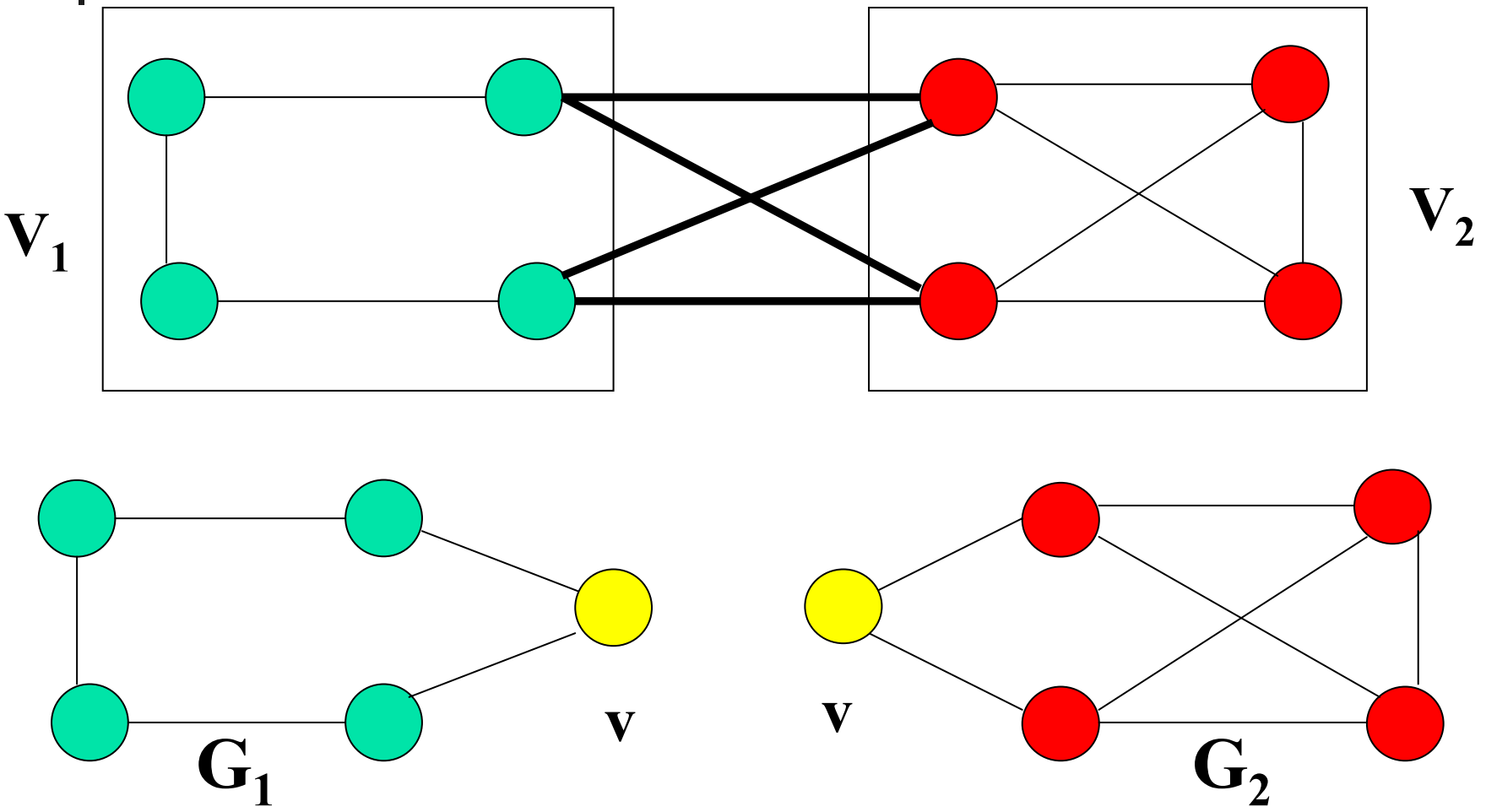


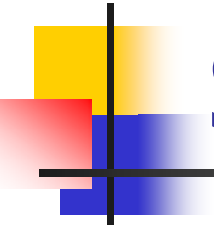
First Split-component



Second Split-component

Example of a split decomposition of a graph



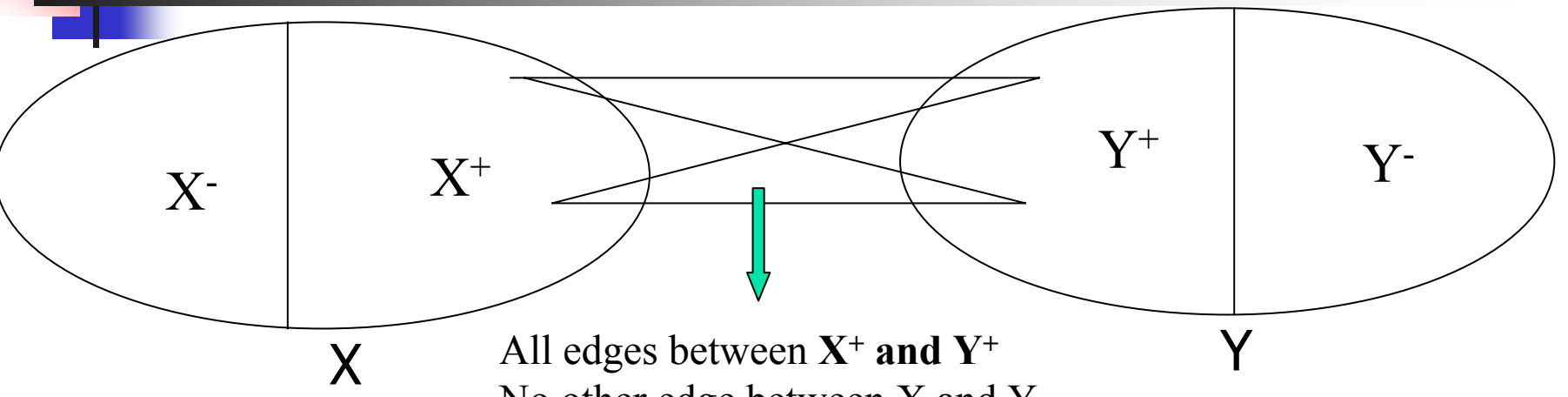


Split decomposition

Definition : The split decomposition of a graph is obtained by recursively decomposing the split-componnents of a graph until obtained prime graphs (i.e. graphs having no split)

The split decomposition is an extension of modular decomposition.

Module of a graph G



If $X^- = \emptyset$ or $Y^- = \emptyset$ then X or resp. Y is a non trivial module of G

The notion of module

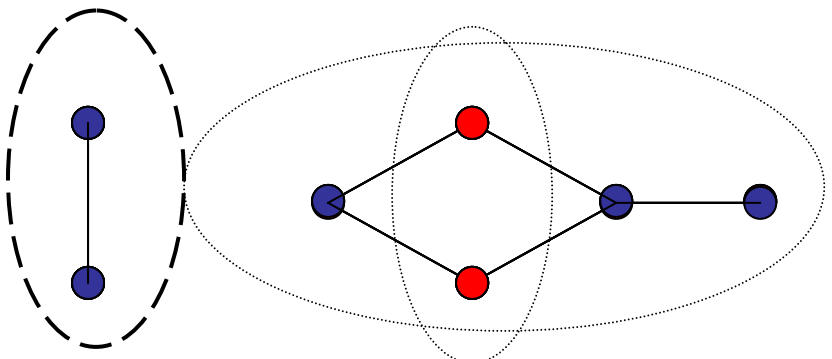
Module : Let G=(V,E) be graphe. $M \subseteq V$ is a module of G iff:

for $x \in V-M$, x is adjacent to every vertex of M or x is adjacent to no vertex of M

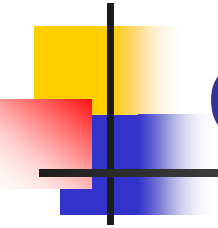
Trivial modules: $\varnothing$, V and the singletons of G

Prime graph:has only trivial modules

Example: a decomposable graph



M is a module of G and co-



Split and modular decomposition of a graph

The importance of these two decompositions motivated many researching works and we dispose linear time algorithms for them.

Distance hereditary graphs

Definition : The distance between two vertices x and y, denoted by $d_G(x,y)$ is the length of a shortest path between x and y

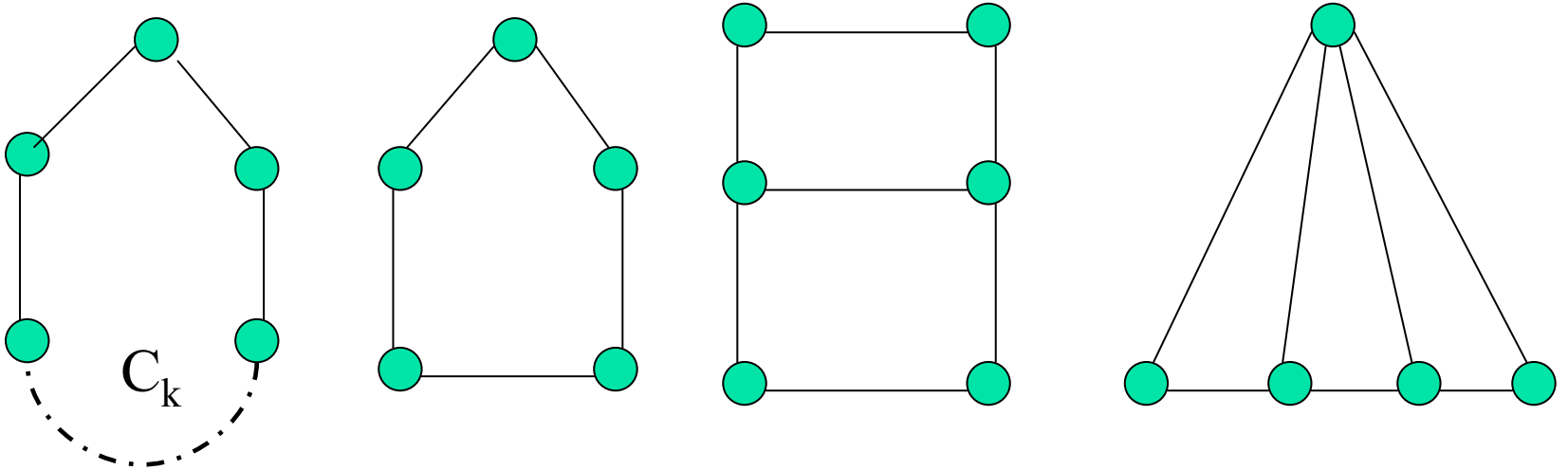
Definition : A graph G is distance hereditary (DH for short) iff for every connected subgraph H of G and every pair (x,y) of vertices $d_H(x,y) = d_G(x,y)$

Important characterisation

G is DH iff G is totaly decomposable by split decomposition

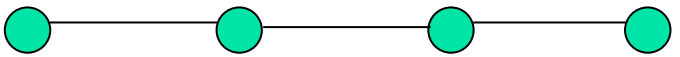
Distance hereditary graphs and forbidden subgraphs

Theorem : A graph G is DH iff G is **DHHG**-free



Cographs and DH graphs

A P_4 is a chordless path with four vertices



A P_4 is self-complemented

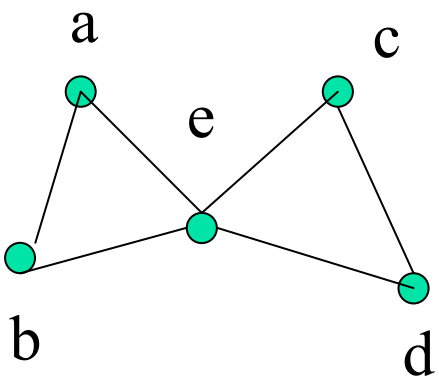
A **cograph** is a graph which is P_4-free.

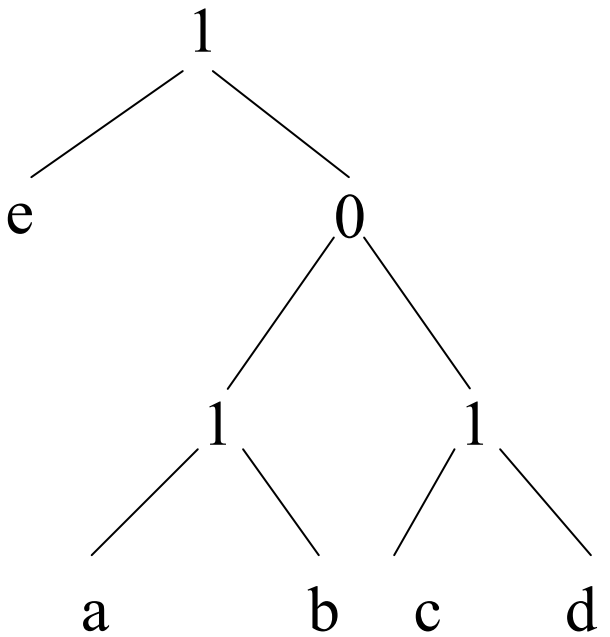
Hence G is a cograph iff co-G is a cograph

The class of cographs is totaly decomposable by modular decomposition.

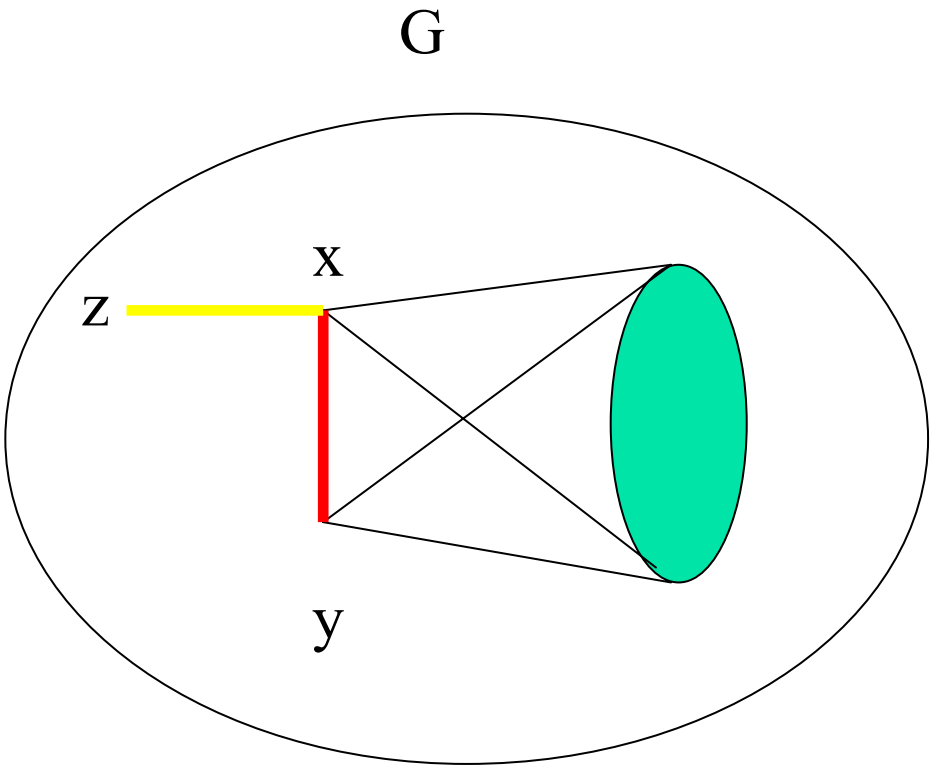
Cotree

A **cotree** is a tree obtained by applying modular decomposition to a cograph





Other caracterization of DH graphs



X and y are

False twins

True twins

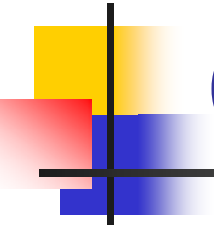
z is a pendant vertex

Second caracterization of DH

Definition : A **pruning sequence** is a total ordering of the vertices $[x_1,\ldots,x_n]$ and a sequence $S=[s_1,\ldots,s_{n-1}]$ of triples, such that for $1 \leq i \leq n-1$, s_i is the one of following words :

- (x_i, P, x_j), x_i is a pendant vertex in G_i
- (x_i, F, x_j), x_i and x_j are false twins in G_i
- (x_i, T, x_j), x_i and x_j are true twins in G_i

Theorem : G is DH iff it has a pruning sequence.



Co-Distance Hereditary graphs

Definition : A graph G is co-DH iff co-G is DH.

We shall give a linear time recognition algorithm for G.

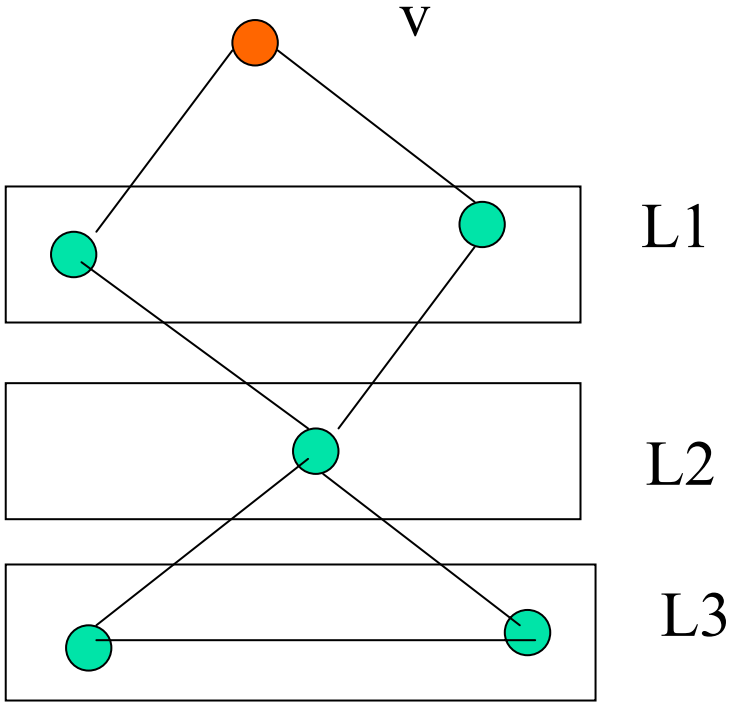
Our algorithm will test if co-G is a DH graph. For this we shall

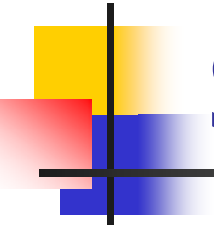
use the recognition algorithm of Damiand, Habib and Paul. There are **6 steps** in this algorithm and we shall show that we did not need to compute co-G for all of these steps but we can work on G and make the necessary transformations in order to remain in linear time on the size of G

STEP 1: Build the distance levels from a vertex v of co-G

$\{L_1,...L_k\}$ is the set of vertices of

co-G such that x belongs to L_i if $d_{co\text{-}G}(v,x)=i$.





STEP 1: distance levels in co-G

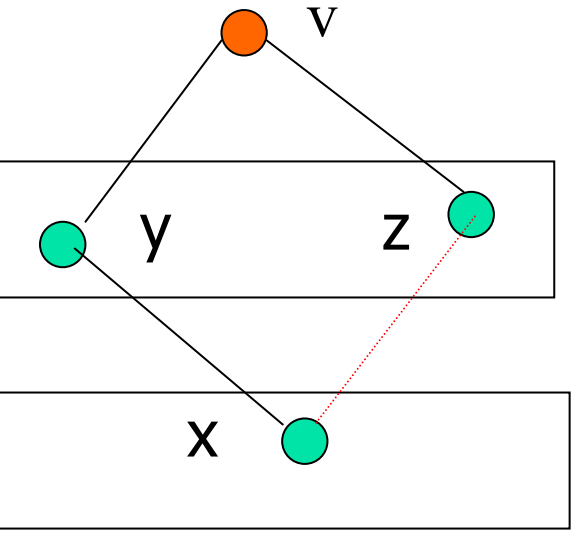
We dispose a list L of all vertices of G and an array A such that A[i]=direct acces to i in L

We compute L_1 from G by
-deleting from L every neighboor x of v
-inserting x in a new liste N_1
-updating A[x] -> direct acces to x in N_1.

L_1 will be formed by the remaining vertices of L

STEP 1: distance levels in co-G

L_2 will be formed by the vertices of N_1 which are not adjacent to all vertices of L_1.

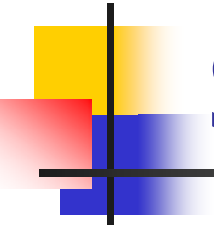


L_1

L_2

For finding L_2 we use an array Index[1..n], initialized to 0.
We visit every neighboor of every vertex in L_1.
We increase by 1 index[z] for every neighboor z in N_1. If Index[z]= $|L_1|$ then we remove z from N_1 and we insert it to a new list N_2

L_2 will be the set of the remaining vertices of N_1



STEP 1

We proceed in an analogous way for computing the other distance levels in co-G.

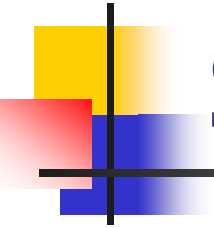
Note that if every vertex of N_i is adjacent to every vertex of L_i then $\{v\} \cup L_1 \cup \ldots \cup L_i$ is a connected component of co-G. In this case we re-start the computation of distance Levels from a new vertex u of N_i

STEP 2

Construct all connected components of every distance level L_i (we know that the corresponding graph is a cograph).

Each $G[L_i]$ is the cograph. iff co-$G[L_i]$ is a cograph

Construct a cotree of $G[L_i]$ and change 0-nodes into 1-nodes and 1-nodes in to 0-nodes. We find in this way the connected components of co-$G[L_i]$



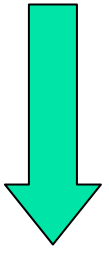
STEP 3

Construct a prunning sequence of co-G[L_i]

Using the algorithm of Damiand, Habib and Paul (2001) we can construct a prunning sequence of a cograph in linear time using the corresponding cotree. Once the pruning sequence is constructed, the algorithm contracts each connected component to the last vertex of this pruning sequence.

STEP 4

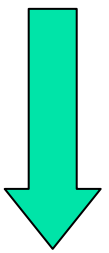
Sort the vertices of Li by increasing inner degree



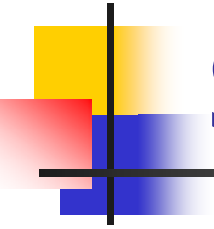
Sort the vertices L_i by decreasing inner degree

STEP5

For every vertex x of L_i having exactly one neighboor y in L_{i-1}: insert (xPy) in pruning sequence



If x has exactly one neighboor in L_i in the graph co-G, x has $|L_{i-1}|$ neighboors in G.



STEP 6

For every vertex x of L_i taken in increasing degree order, construct a pruning sequence of the graph induced by $[N_{i-1}(x)]$.
(***We know that it is a cograph***)

Once the vertices of L_i have been sorted by decreasing inner order, using the array A we can find the non-neighborhood in L_{i-1} of each vertex x in L_i with in O(degree(x)) complexity.

We then proceed as described on Steps 2 and 3

K-expression

[Courcelle, Engelfriet, Rozenberg, 93]

- **k-expression : Expression allowing to construct a graph using at most k labels and the following operations:**

  i(v): label the vertex v by the label i

  $\oplus$ disjoint union of two graphs

  $\eta_{i \to j}$ Connecting all vertices labeled i with them labeled j

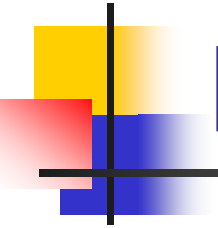
  $\rho_{i \to j}$ Change labels i into labels j

Clique width

- Is the minimum number of labels needed for defining a graph by a k-expression

: [Courcelle, Makowski, Rotics, 98]

There exists efficient solutions for many optimization problems for any class of graphs whose clique-width is bounded by a constant k.



K-expression for DH graphs

Golumbic and Rotics (2000) proposed a 3-expression for DH graphs.

This expression was obtained by contructing from the Pruning sequence of a DH graph G a special tree T(G) the **pruning tree**.

Constructing the 3-expression for DH graphs

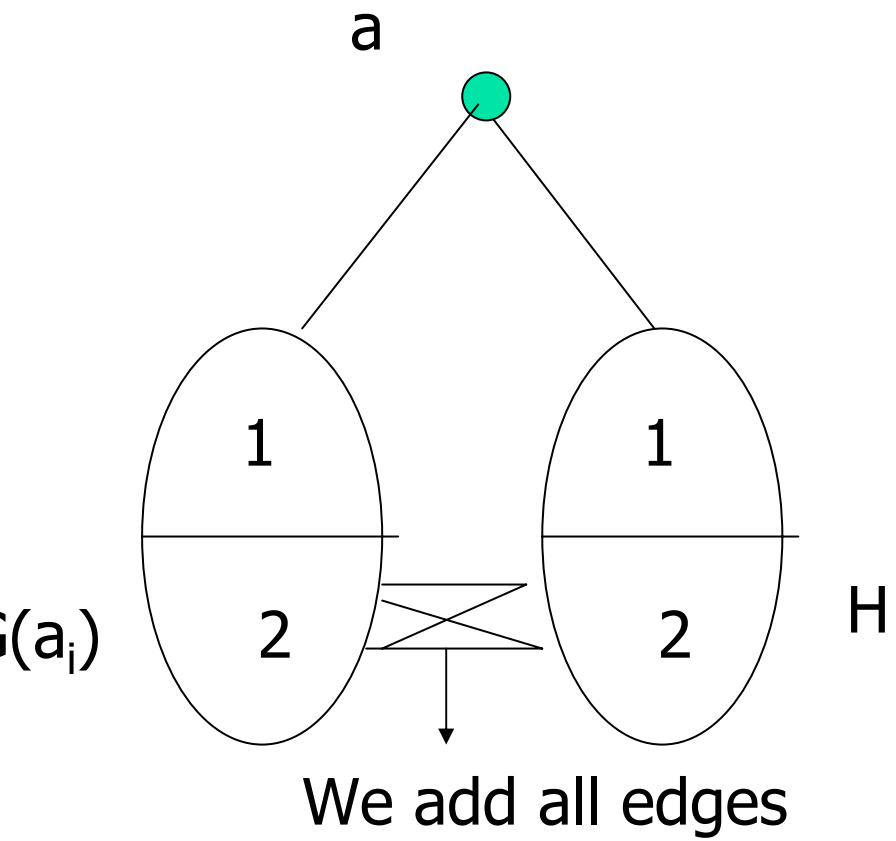
Let a be an internal node of the pruning tree and $a_1,\ldots,a_l$ its sons from left to right. We assume that we know the k-expressions for $G(a_i)$ and $H=\{ G(a_{i+1}) \mathbf{U} \ldots \mathbf{U} G(a_l) \}$

for constructing the graph $G(a_i) \mathbf{U} H$ either

1. We consider the union of these two graphs
2. Or we add edges between $G(a_i)$ and H joining

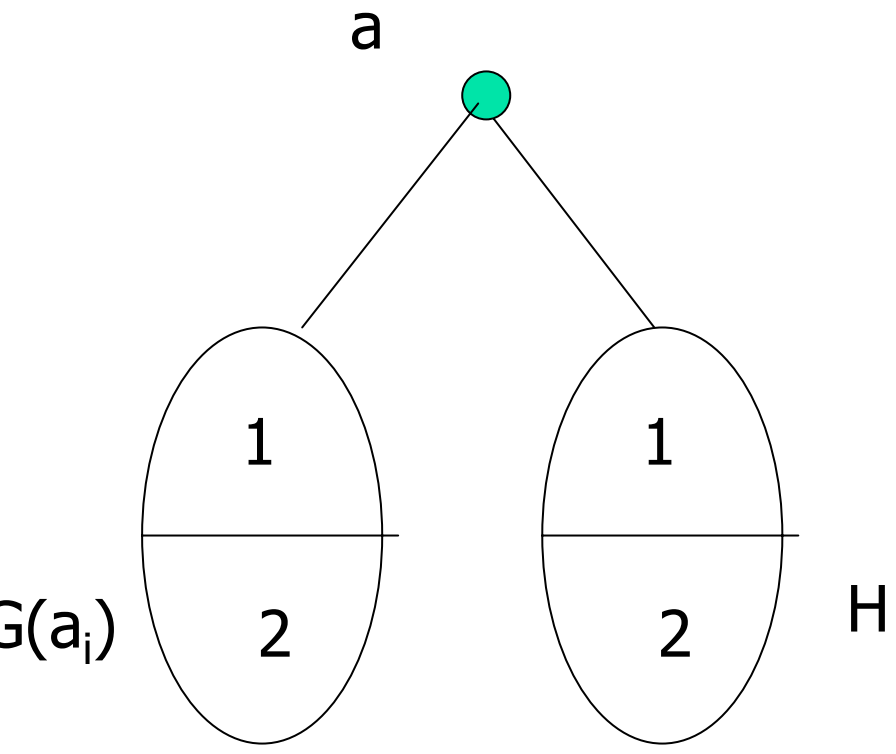
a special kind of descendants of a (twin descendants). These descendants are labeled 2 and all the other vertices 1.

Constructing the 3-expression for DH graphs



For avoiding to add edges between the vertices labeled
2 in the same graph, we use
the label 3 for renaming
the vertices labeled 2 in H,
Then we add all edges between
vertices labeled 2 and 3 and finaly we re-change the label of the vertices labeled 3

Constructing the 4-expression for co-DH graphs



Either we will have to add edges between all vertices of $G(a_i)$ and H or we add all edges except those Concerning vertices labeled 2.
In any case we need 4 labels, we give the label 3 to the vertices labeled 1 in H, the label 4 to the vertices labeled 2 in H before adding the necessary edge between $G(a_i)$ and H.

THANKS YOU