# 7th Lecture : Graph searches and graph classes (perfect phylogeny) MPRI 2013–2014

Michel Habib
habib@liafa.jussieu.fr
http://www.liafa.jussieu.fr/~habib

Sophie Germain, 5 novembre 2013



## Schedule





## Subtrees in a tree

Using results of Dirac 1961, Fulkerson, Gross 1965, Buneman 1974, Gavril 1974 and Rose, Tarjan and Lueker 1976 :

For a connected graph, the following statements are equivalent and characterize chordal graphs :

(i) G has a simplicial elimination scheme
(ii) Every minimal separator is a clique
(iii) $G$ admits a maximal clique tree.
(iv) $G$ is the intersection graph of subtrees in a tree.
(v) Any MNS (LBF, LexDFS, MCS) provides a simplicial elimination scheme.



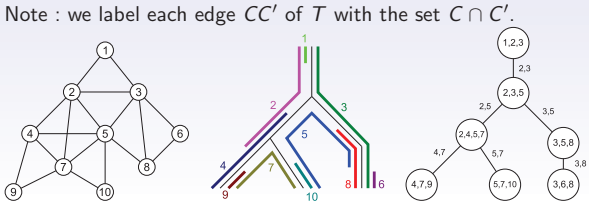
## Clique tree

*clique tree* of $G$ = a minimum size tree model of $G$

for a clique tree $T$ of $G$ :

- vertices of $T$ correspond precisely to the maximal cliques of $G$
- for every maximal cliques $C, C'$, each clique on the path in $T$ from $C$ to $C'$ contains $C \cap C'$
- for each edge $CC'$ of $T$, the set $C \cap C'$ is a *minimal separator* (an inclusion-wise minimal set separating two vertices)

Note : we label each edge $CC'$ of $T$ with the set $C \cap C'$.






## Consequences of maximal clique tree

**Theorem**
Every minimal separator belongs to every maximal clique tree.

**Lemma**
Every minimal separator is the intersection of at least 2 maximal cliques of $G$.

**Corollary**
There are at most $n$ minimal separators.



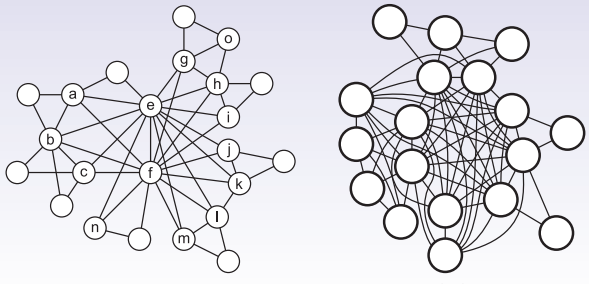
**Theorem**
Every minimal separator belongs to every maximal clique tree.

**Lemma**
Every minimal separator is the intersection of at least 2 maximal cliques of $G$.



## Clique graph

the *clique graph* $\mathcal{C}(G)$ of $G$ = intersection graph of maximal cliques of $G$

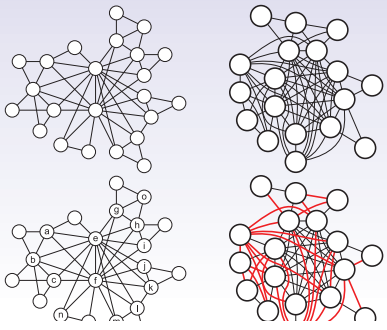


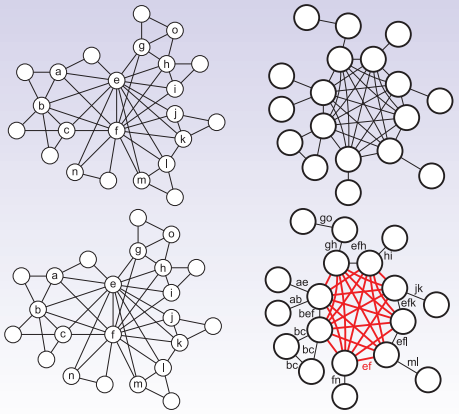

$G$ $\quad$ $\mathcal{C}(G)$



## Reduced clique graph

the *reduced clique graph* $\mathcal{C}_r(G)$ of $G$ = graph on maximal cliques of $G$ where $CC'$ is an edge of $\mathcal{C}_r(G) \iff C \cap C'$ is a minimal separator.








## Combinatorial structure of $\mathcal{C}_r(G)$

**Lemma 1 : M.H and C. Paul 95**

If $C_1, C_2, C_3$ is a cycle in $\mathcal{C}_r(G)$, with $S_{12}, S_{23}$ and $S_{23}$ be the associated minimal separators then two of these three separators are equal and included in the third.

**Lemma 2 : M.H. and C. Paul 95**

Let $C_1, C_2, C_3$ be 3 maximal cliques, if $C_1 \cap C_2 = S_{12} \subset S_{23} = C_2 \cap C_3$ then it yields a triangle in $\mathcal{C}_r(G)$

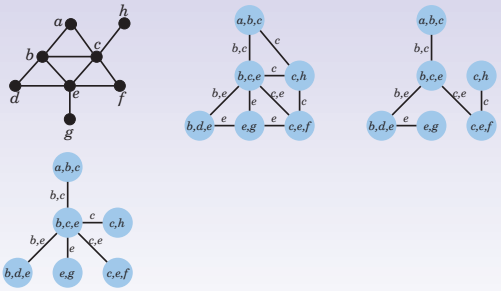


**Lemma 3 : Equality case**

Let $C_1, C_2, C_3$ be 3 maximal cliques, if $S_{12} = S_{23}$ then :

- either the $C_1 \cap C_3 = S_{13}$ is a minimal separator
- or the edges $C_1C_2$ and $C_2C_3$ cannot belong together to a maximal clique tree of $G$.








**Theorem (Gavril 87, Shibata 1988, Blayr and Payton 93)**

*The clique trees of $G$ are precisely the maximum weight spanning trees of $\mathcal{C}(G)$ where the weight of an edge $CC'$ is defined as $|C \cap C'|$.*

**Theorem (Galinier, Habib, Paul 1995)**

*The clique trees of $G$ are precisely the maximum weight spanning trees of $\mathcal{C}_r(G)$ where the weight of an edge $CC'$ is defined as $|C \cap C'|$.*

*Moreover, $\mathcal{C}_r(G)$ is the union of all clique trees of $G$.*



## Applications

- All clique trees have exactly the same labels, including repetitions.



## Maximal Cardinality Search : MCS

**Data**: a graph $G = (V, E)$ and a start vertex $s$
**Result**: an ordering $\sigma$ of $V$
Assign the label 0 to all vertices
$label(s) \leftarrow 1$
**for** $i \leftarrow n$ **à** 1 **do**
  Pick an unumbered vertex $v$ with largest label
  $\sigma(i) \leftarrow v$
  **foreach** *unnumbered vertex $w$ adjacent to $v$* **do**
    $label(w) \leftarrow label(w) + 1$
  **end**
**end**



**Maximum spanning trees**

Maximal Cardinality Search can be seen as Prim algorithm for computing a maximal spanning tree of $\mathcal{C}_r(G)$.

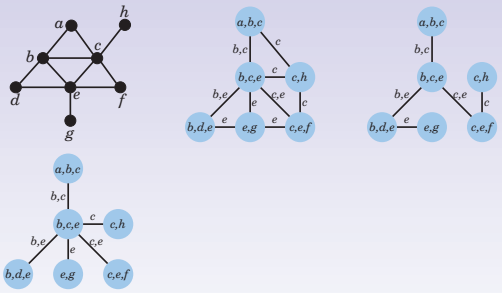
- How to compute a clique tree ?
- How to generate all simplicial elimination schemes ?



LBF, MCS or MNS visit maximal cliques "consecutively" (i.e. when the search explores a vertex $x$ of a maximal clique $C$ that does not belong to any of the previously visited maximal cliques then all the unvisited vertices of $C$ will appear consecutively just after $x$).

Therefore when applying a search (LBFS, MCS or MNS) one can compute a clique tree, by considering the strictly increasing sequences of labels.






$b, a, c, e, d, h, f, g$ is a LBFS ordering.
we can find the maximal cliques $b, a, c$ then $b, c, e$ then $b, e, d$ then $c, h$ then $c, e, f$ and $e, g$.



## Simplicial elimination schemes

1. Choose a maximal clique tree $T$
2. While $T$ is not empty do
   Select a vertex $x \in F - S$ in a leaf $F$ of $T$ ;
   $F \leftarrow F - x$ ;
   If $F = S$ delete $F$ ;



## Canonical simplicial elimination scheme

1. Choose a maximal clique tree $T$
2. While $T$ is not empty do
   Choose a leaf $F$ of $T$ ;
   Select successively all vertices in $F - S$
   delete $F$ ;



## Remark

Does there exist other simplicial elimination scheme ?



## Size of a maximal clique tree in a chordal graph

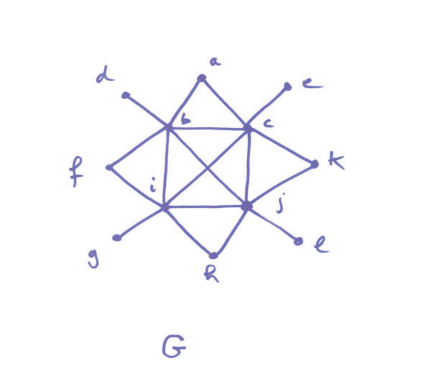
- Let $G = (V, E)$ be a chordal graph.
- $G$ admits at most $|V|$ maximal cliques and therefore the tree is also bounded by $|V|$ (vertices and edges).
- But some vertices can be repeated in the cliques. If we consider a simplicial elimination ordering the size of a given maximal clique is bounded by the neighbourhood of the first vertex of the maximal clique.
- Therefore any maximal clique tree is bounded by $|V| + |E|$.



## Size of $\mathcal{C}_r(G)$

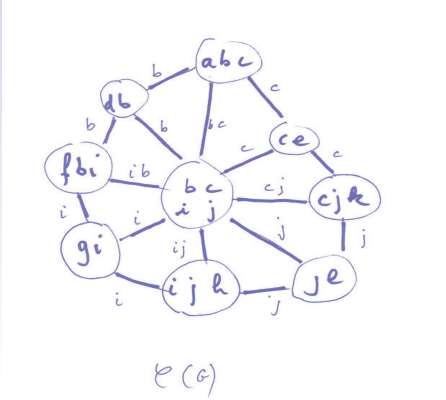
Considering a star on n vertices,
shows $|\mathcal{C}_r(G)| \in O(n^2)$
Not linear in the size of $G$

## $\mathcal{C}_r(G)$ is not chordal !






## $\mathcal{C}_r(G)$ is not chordal !





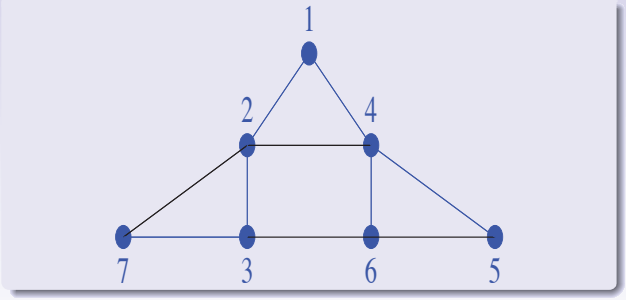

In fact $\mathcal{C}_r(G)$ is dually chordal (almost chordal) and $\mathcal{C}_r(\mathcal{C}_r(G))$ is chordal.







## Generic Search




**Invariant**

At each step, an edge between a visited vertex and a unvisited one is selected



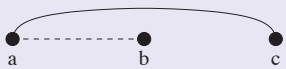
**Generic search**

$S \leftarrow \{s\}$
**for** $i \leftarrow 1$ **to** $n$ **do**
  Pick an unumbered vertex $v$ of $S$
  $\sigma(i) \leftarrow v$
  **foreach** *unumbered vertex* $w \in N(v)$ **do**
    **if** $w \notin S$ **then**
      Add $w$ to $S$
    **end**
  **end**
**end**



**Generic question ?**

Let $a$, $b$ et $c$ be 3 vertices such that $ab \notin E$ et $ac \in E$.

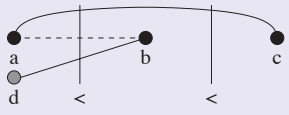



Under which condition could we visit first $a$ then $b$ and last $c$ ?



**Property (Generic)**

For an ordering $\sigma$ on $V$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $d < b$ et $db \in E$




**Theorem**

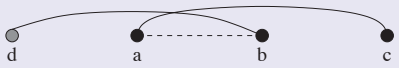
For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a generic search of $G$ iff $\sigma$ satisfies property (Generic).

Most of the searches that we will study are refinement of this generic search
i.e. we just add new rules to follow for the choice of the next vertex to be visited
Graph searches mainly differ by the management of the tie-break set



### Property (BFS)

For an ordering $\sigma$ on $V$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $d < a$ et $db \in E$



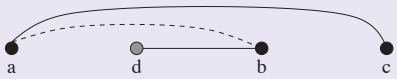

### Theorem

For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a BFS of $G$ iff $\sigma$ satisfies property (BFS).



### Proprerty (DFS)

For an ordering $\sigma$ on $V$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $a < d < b$ and $db \in E$.




### Theorem

For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a DFS of $G$ iff $\sigma$ satisfies property (DFS).



## Applications of BFS

1. Distance computations (unit length), diameter and centers
2. BFS provides a useful layered structure of the graph
3. Using BFS to search an augmenting path provides a polynomial implementation of Ford-Fulkerson maximum flow algorithm.



## Applications of DFS

### Some applications

- Planarity testing.
- Computation of 2-connected (resp. strongly connected) components, 2-SAT solvers
- Computation of a linear extension (topological sorting) for an acyclic digraph, applications to inheritance mechanisms. ...



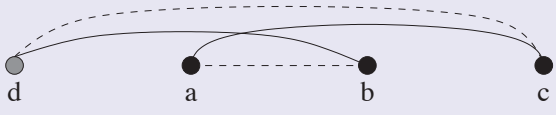
## Lexicographic Breadth First Search (LBFS)

**Data**: a graph $G = (V, E)$ and a start vertex $s$
**Result**: an ordering $\sigma$ of $V$
Assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{n\}$
**for** $i \leftarrow n$ **to** 1 **do**
  Pick an unumbered vertex $v$ with lexicographically largest label
  $\sigma(i) \leftarrow v$
  **foreach** *unnumbered vertex w adjacent to v* **do**
    $label(w) \leftarrow label(w).\{i\}$
  **end**
**end**



### Property (LexB)

For an ordering $\sigma$ on $V$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $d < a$ et $db \in E$ et $dc \notin E$.




### Theorem

For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a LBFS of $G$ iff $\sigma$ satisfies property (LexB).



## Why LBFS behaves so nicely on well-structured graphs

A nice recursive property :
On every tie-break set $S$, LBFS operates on $G(S)$ as a LBFS.
Analogous properties are false for other classical searches.

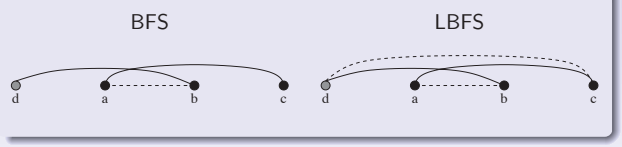
## Applications of LBFS

1. Most famous one : Chordal graph recognition
2. For many classes of graphs using LBFS ordering "backward" provides structural information on the graph.
3. Last visited vertex (or clique) has some property (example simplicial for chordal graph)

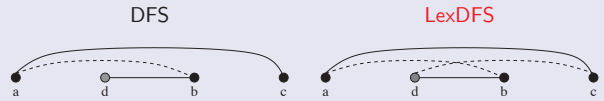


## LDFS Lexicographic Depth First Search

### BFS vs LBFS



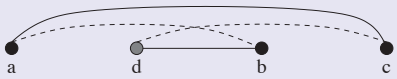

### DFS vs LexDFS






### Property (LDFS)

For an ordering $\sigma$ on $V$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex $d$ such that $a < d < b$ and $db \in E$ and $dc \notin E$.




### Theorem

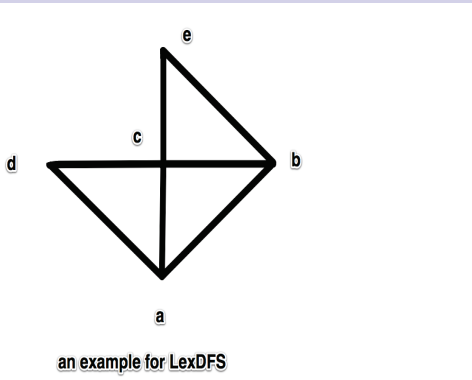
For a graph $G = (V, E)$, an ordering $\sigma$ sur $V$ is a LDFS of $G$ iff $\sigma$ satisfies property (LDFS).



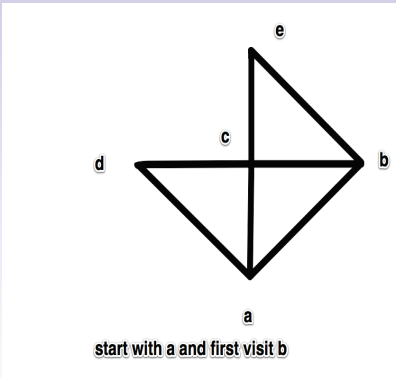
## Lexicographic Depth First Search (LexDFS)

**Data**: a graph $G = (V, E)$ and a start vertex $s$
**Result**: an ordering $\sigma$ of $V$
Assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{0\}$
**for** $i \leftarrow 1$ **to** $n$ **do**
  Pick an unumbered vertex $v$ with lexicographically largest label
  $\sigma(i) \leftarrow v$
  **foreach** *unnumbered vertex w adjacent to v* **do**
    $label(w) \leftarrow \{i\}.label(w)$
  **end**
**end**

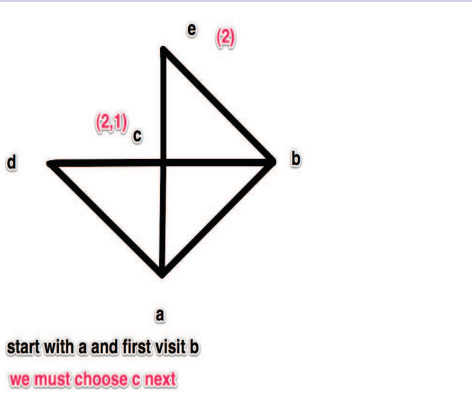




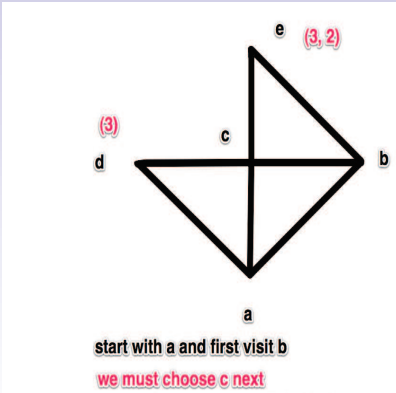


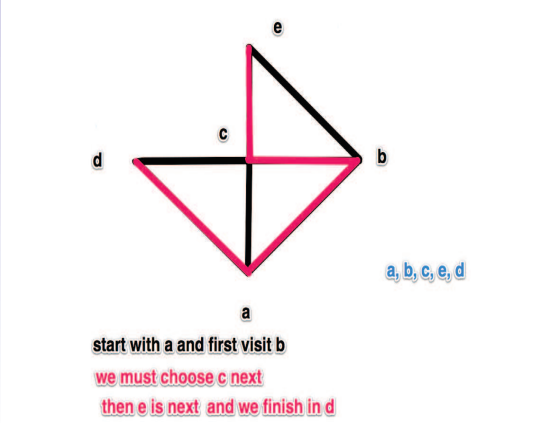












## LDFS

**Complexity**

Is it possible to compute a LDFS in $O(n+m)$ ?

Spinrad, 2008 Best implementation so far needs $O(n + mloglogn)$ using Van der Boas trees.

Mouatadid, Kőhler, 2013 Linear time implementation on cocomparability graphs (using partition refinement).

**First application : D. Corneil, B. Dalton, M. H. 2013**

Hamiltonicity on cocomparability graphs via LDFS.



## Maximal Neighbourhood Search (MNS)

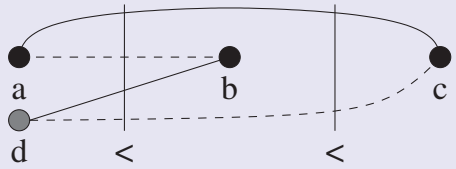
**MNS**

**Data**: a graph $G = (V, E)$ and a start vertex $s$
**Result**: an ordering $\sigma$ of $V$
Assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{0\}$
**for** $i \leftarrow 1$ **to** $n$ **do**
  Pick an unumbered vertex $v$ with a maximal under inclusion label
  $\sigma(i) \leftarrow v$
  **foreach** *unnumbered vertex w adjacent to v* **do**
    $label(w) \leftarrow \{i\} \cup label(w)$
  **end**
**end**



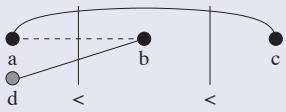
**MNS property**

Let $\sigma$ be a total ordering $V(G)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it exsits $d$ such that $d < b$, $db \in E$ and $dc \notin E$.






**Generic search**




**MNS**

MNS is a kind of completion of Generic search similar to BFS versus LBFS (resp. DFS versus LDFS). This explains why MNS was first named LexGen.



**Theorem [Tarjan et Yannakakis, 1984]**

$G$ is a chordal graph iff every MNS computes a simplicial ordering.

**Proof :**

Let $c$ be a non simplicial vertex (to the left). Thus it exists $a < b < c \in N(c)$ with $ab \notin E$. Using MNS property, it exsits $d < b$ with $db \in E$ and $dc \notin E$. Since $G$ is chordal, necessarily $ad \notin E$.
Either $d < a$, considering the triple $d, a, b$, it exists $d' < a$ such that $d'a \in E$ and $d'b \notin E$. Furthermore $d'd \notin E$.
Or $a < d$, considering the triple $a, d, c$, it exists $d' < d$ such that $d'd \in E$ and $d'c \notin E$. Furthermore $ad' \notin E$.
In both cases a pattern is propagating to the left, a contradiction.



**Corollary**

$G$ is a chordal graph iff every MCS, LBFS, LDFS computes a simplicial ordering.

**Proof**

Maximal for the cardinality, or maximal lexicographically are particular cases of maximality under inclusion.

**Implementation**

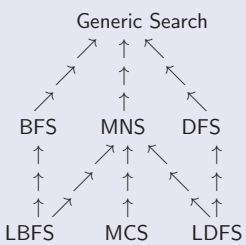
MCS, LBFS provide linear time particular implementation sof MNS. But they are many others, less famous.
But in its full generality no linear time implementation is known.



## Conclusions

Using the 4-points configurations we can prove the following inclusion ordering between searches

**Strict inclusions**



## Search classification

| Search | Tie-break management |
|---|---|
| Generic search | none (random) |
| BFS | queue |
| DFS | stack |
| LBFS | Lexicographic maximal |
| LDFS | Lexicographic maximal |
| MNS | Maximal under inclusion |
| MCS | Maximal for the cardinality |



## Applications

- BFS to compute distances, diameter, centers
  Heuristics for diameter
- DFS planarity, strongly connected components, 2-SAT, ...
- LBFS, recognition of chordal graphs, interval graphs ...
  Recursive behavior on tie-break sets.
  Heuristics for one consecutiveness property
- LDFS, long paths, minimum path cover
  For cocomparability graphs LDFS computes layered ordering of the complement partial order.
  Heuristics for graph clustering, still many applications to be discovered.



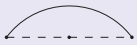
**Definition**

A cocomparability graph $G$ is the complement of a comparability graph, i.e., $\overline{G}$ admits a transitive orientation.

**Definition :**

For a total ordering $\tau$ of the set of vertices, an umbrella is a triple of vertices $a, b, c \in X$ such that : $a <_\tau b <_\tau c$ and $ac \in E$ and $ab, bc \notin E$. A cocomparability (cocomp for short) ordering is an umbrella-free total ordering of the vertices of $G$.

**Forbidden triple**




An umbrella



## Characterization Theorem for interval graphs

Gilmore and Hoffman 1964

(i) $G$ is an interval graph, i.e. it can be represented as the intersection graph of a family of intervals of the real line.

(ii) It exists a total ordering $\tau$ 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 $xy \in E$. An interval ordering.

(iii) G has a maximal clique path. (A maximal clique path is just a maximal clique tree T, reduced to a path).

(iv) G is chordal and cocomparability.



## Proof

(i)→(ii) For $\tau$ we take the ordering of vertices obtained by the ordering of the leftcorners of the intervals. So $x \leq_\tau y$ iff $left(x) \leq left(y)$. $\forall x, y, z \in G$ with $x \leq_\tau y \leq_\tau z$, every interval $I(y)$ starting between $left(x)$ and $left(z)$ must intersect $I(x)$, if $xz \in E$. Therefore $\tau$ is an ordering satisfying the condition.

(i)→(iii) The maximal cliques can be obtained from the interval representation by a sweep from left to right of a vertical line, which yields a maximal clique path.

(i)→(iv) We already see that every interval graph is a chordal graph. Let us define a relation $R$ on $V(G)$, as follows :
$xRy$ iff $I(x)$ completely to the left of $I(y)$. Clearly $R$ is a partial order and therefore $\overline{G}$ is a comparability graph.



## Proof II

(ii)→(i) To each vertex $x$ we construct $I(x) = [x, z]$ where $z$ is the rightmost neighbour of $x$ in $\tau$. Using the property of $\tau$, we know that $\forall y \in [x, z]$, $xy \in E(G)$, and therfore we have an interval representation of $G$.

(iii)→(i) From a maximal clique path, it is very easy to construct the interval representation, by taking for every vertex the list of consecutive maximal cliques that contain it.

(ii)→(iv) First we remark that $\tau^d$ is a simplicial elimination scheme. Furthermore $\tau$ is a cocomp ordering. If it exists $a, b, c \in V(G)$, with $ab, bc \notin E(G)$ and $a \leq_\tau b \leq_\tau c$. Either $I(a), I(b), I(c)$ pairwise do not intersect and therefore $ac \notin E(G)$, or $I(a)$ and $I(c)$ intersect, but then using the defintion of $\tau$ this implies $ab \in E(G)$, a contradiction.



**Lemma 1(Corneil)**

For every cocomparability graph $G$, there exists a LBFS ordering which is a cocomp ordering.

**Lemma 2**

Any simplicial cocomp ordering is an interval ordering.

**proof :**

Let us consider $\tau$ simplicial from right to left which is a cocomp ordering.
Let us consider $\forall x, y, z \in V(G)$ with $x \leq_\tau y \leq_\tau z$ and $xz \in E(G)$. Since $\tau$ is a cocomp ordering, necessarily $xy$ or $yz$ is an edge. If $xy \in E(G)$, we are done. Else if $yz \in E(G)$ then using the simpliciality of $x$, we have $xy \in E(G)$. In both cases $\tau$ is an interval ordering.



## Proof III

**(iv) → (ii)**

Using the previous lemma 1, we can consider a LBFS cocomp ordering $\tau$ of $G$. Since $G$ is chordal, $\tau$ is also simplicial (as any LBFS), and we can use lemma 2 to end the proof.

All the previous proofs play only with orderings of the vertices and 4-points conditions.
The proofs of the next theorem are left as an exercise.



## Characterization theorem for proper interval graphs

Roberts 1970, Wegner 1967, Loodges and Olariu 1993.

(i) $G$ is a proper interval graph, i.e. it can be represented as the intersection graph of a family of intervals of the real line, in which no interval strictly contains another interval.

(ii) It exists a total ordering $\tau$ 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 $xy$ and $yz \in E$. A proper interval ordering.

(iii) G is unit-interval graphs (i.e. it admits a representation with intervals of the same length).

(iv) G is an interval graph and does not contain any $K_{1,3}$.

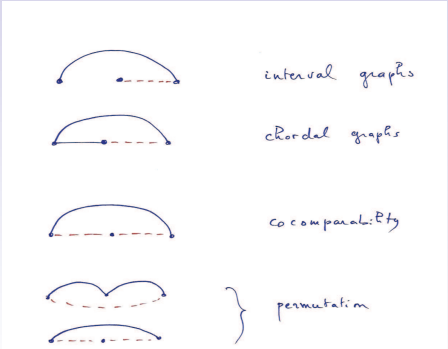


## So far we have seen

Chordal graph iff $\exists$ a simplicial ordering.
Interval graph iff $\exists$ an interval ordering.
Proper interval graph $\exists$ a proper interval ordering.
Cocomparability graph iff $\exists$ a cocomp ordering.



## A description using forbidden configurations






## Consequences

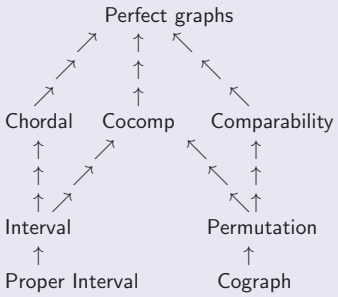
LBFS is involved in many recognition algorithms for these classes of graphs, in fact due to the following property for cocomparability and comparability graphs :

- If $G$ is a cocomparability graph and $\sigma$ a LBFS on $G$, then the last vertex of $\sigma$, can be taken as a source in a transitive orientation of $\overline{G}$.
- This is the starting step for comparability and permutation graph recognition algorithms, using partition refinement.



## A hierarchy of some hereditary classes of graphs

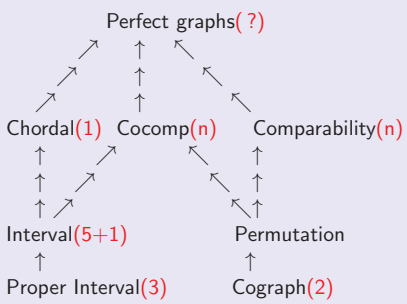
Strict inclusions between classes





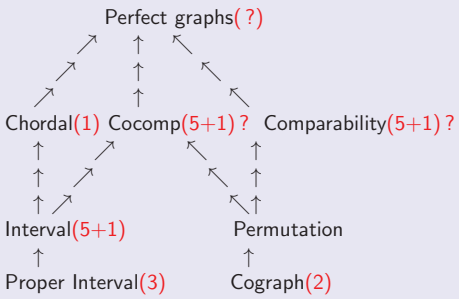

## A hierarchy of some hereditary classes of graphs

Minimum number of LBFS used in a recognition algorithm



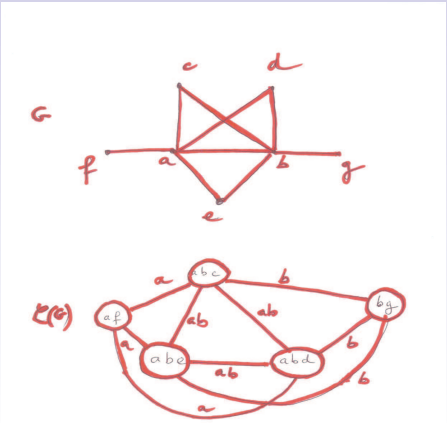



Conjectures

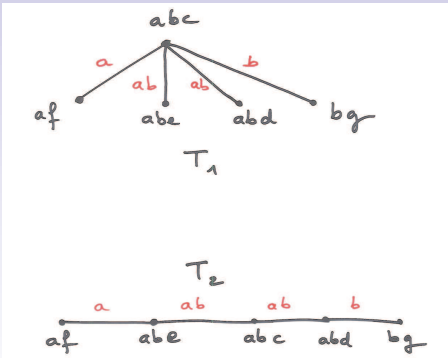


To recognize an interval graph, we just have to compute a maximal clique tree and check if it is a path ?
Difficulty : an interval graph has many clique trees among them some are paths










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

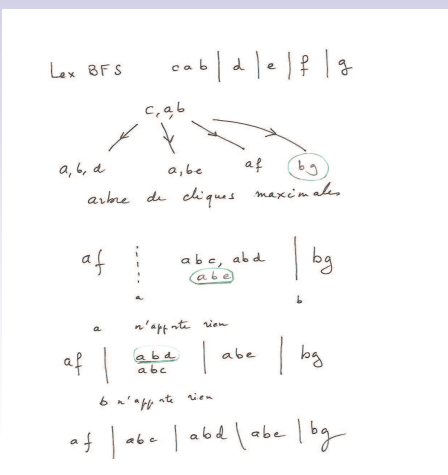
- Booth and Lueker 1976, using PQ-trees.
- Korte and Mohring 1981 using LBFS and Modified PQ-trees.
- Hsu and Ma 1995, using modular decomposition and a variation on Maximal Cardinality Search.
- Corneil, Olariu and Stewart SODA 1998, using a series of 6 consecutive LBFS, published in 2010.
- M.H, McConnell, Paul and Viennot 2000, using LexBFS and partition refinement on maximal cliques.
- P. Li, Y. Wu 2011, using a series of 4 sort of LBFS
- . . .



## A partition refinement algorithm working on maximal cliques

1. Compute a tree $T$ using LBF
   If $T$ is not a maximal clique tree ; then $G$ is not chordal, neither interval.
2. Start form the last maximal clique 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.








## Exercise 1

### Ends of a LexBFS

Many properties can be expressed on the last vertex of a LexBFS.
Example : if $G$ is a chordal graph, the last vertex is simplicial .

1. Show that the last maximal clique visited can be taken as the end of some chain of cliques if $G$ is an interval graph.
2. Complexity of the following decision problem :

**Data**: a graph $G = (V, E)$ and a given vertex $x \in V$
**Result**: Does there exist a LBFS of $G$ ending at $x$ ?

## Exercise 2

If we consider the edges of the clique tree labelled with the size of the minimal separators, show that :
for every maximal clique tree $T$
$weight(T) = \Sigma_{1 \leq i \leq k} |C_i| - n$, where $C_1, \ldots C_k$ are the maximal cliques of $G$.