Séminaire MDOD du CES – Université Paris-1
29/05/2012 - Paris

Codage des voisinages et parcours des graphes d'intervalles et de permutation

Christophe Crespelle
LIP
Université Claude-Bernard
Lyon 1

Philippe Gambette
LIGM
Université Paris-Est
Marne-la-Vallée

Plan

- Graphes d'intersection
- Algorithme de parcours en largeur avec priorités
- Codage des voisinages
- Contiguïté et linéarité

Plan

- Graphes d'intersection
- Algorithme de parcours en largeur avec priorités
- Codage des voisinages
- Contiguïté et linéarité

Les graphes d'intersection

un **sommet** ⇔ un **objet**

une **arête** entre deux sommets ⇔ les deux objets ont une **intersection non vide**

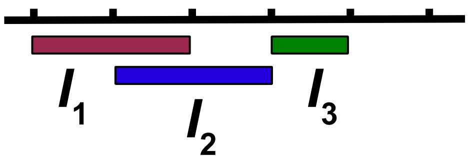
$\mathcal{I}$ est une **réalisation** du **graphe d'intersection** *G.*

G est un **graphe d'intersection.**

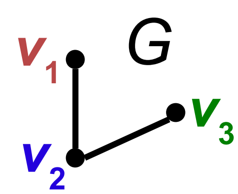
Les graphes d'intervalles

$\mathcal{I} = \{[0,2],[1,3],[3,4]\}$

un **sommet** ⇔ un **intervalle**



une **arête** entre deux sommets ⇔ les deux intervalles ont une **intersection non vide**



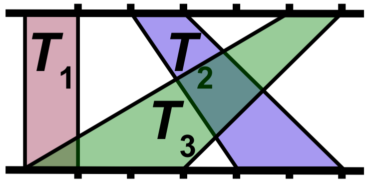
$\mathcal{I}$ est une **réalisation intervallaire** de *G*.

G est un **graphe d'intervalles.**

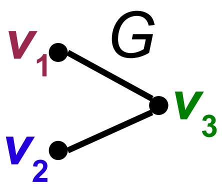
Les graphes trapézoïdaux

$$\mathcal{T} = \{([0,1],[0,1]),\ ([2,3],[4,6]),\ ([5,6],[0,3])\}$$

un **sommet** ⇔ un **trapèze** entre deux lignes horizontales



une **arête** entre deux sommets ⇔ les deux intervalles ont une **intersection non vide**



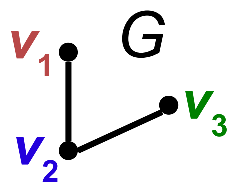
G est un **graphe trapézoïdal.**

Classes de graphes

Tout graphe d'intervalles est un graphe trapézoïdal.

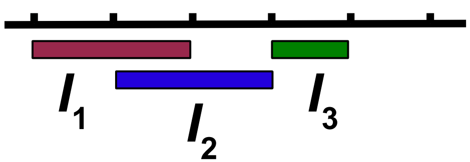
Classes de graphes

Tout graphe d'intervalles est un graphe trapézoïdal.



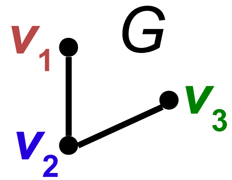
G graphe d'intervalles

$\mathcal{I} = \{[0,2],[1,3],[3,4]\}$



Classes de graphes

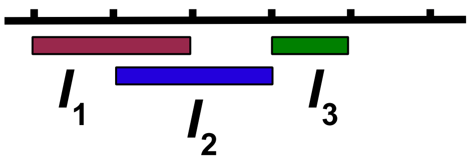
Tout graphe d'intervalles est un graphe trapézoïdal.

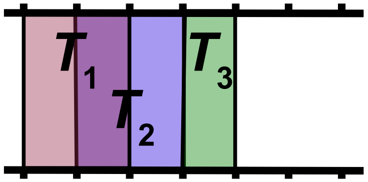


$$\mathcal{I} = \{[0,2],[1,3],[3,4]\}$$

transformation de la réalisation

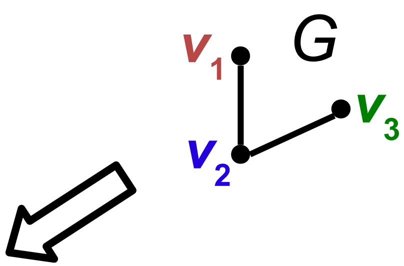
$$\mathcal{T} = \{([0,2],[0,2]),([1,3],[1,3]),([3,4],[3,4])\}$$



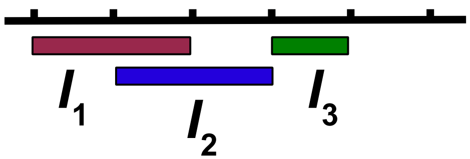


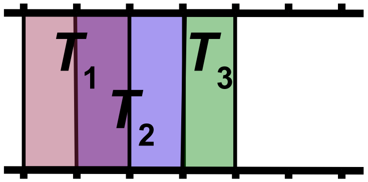
Classes de graphes

Tout graphe d'intervalles est un graphe trapézoïdal.



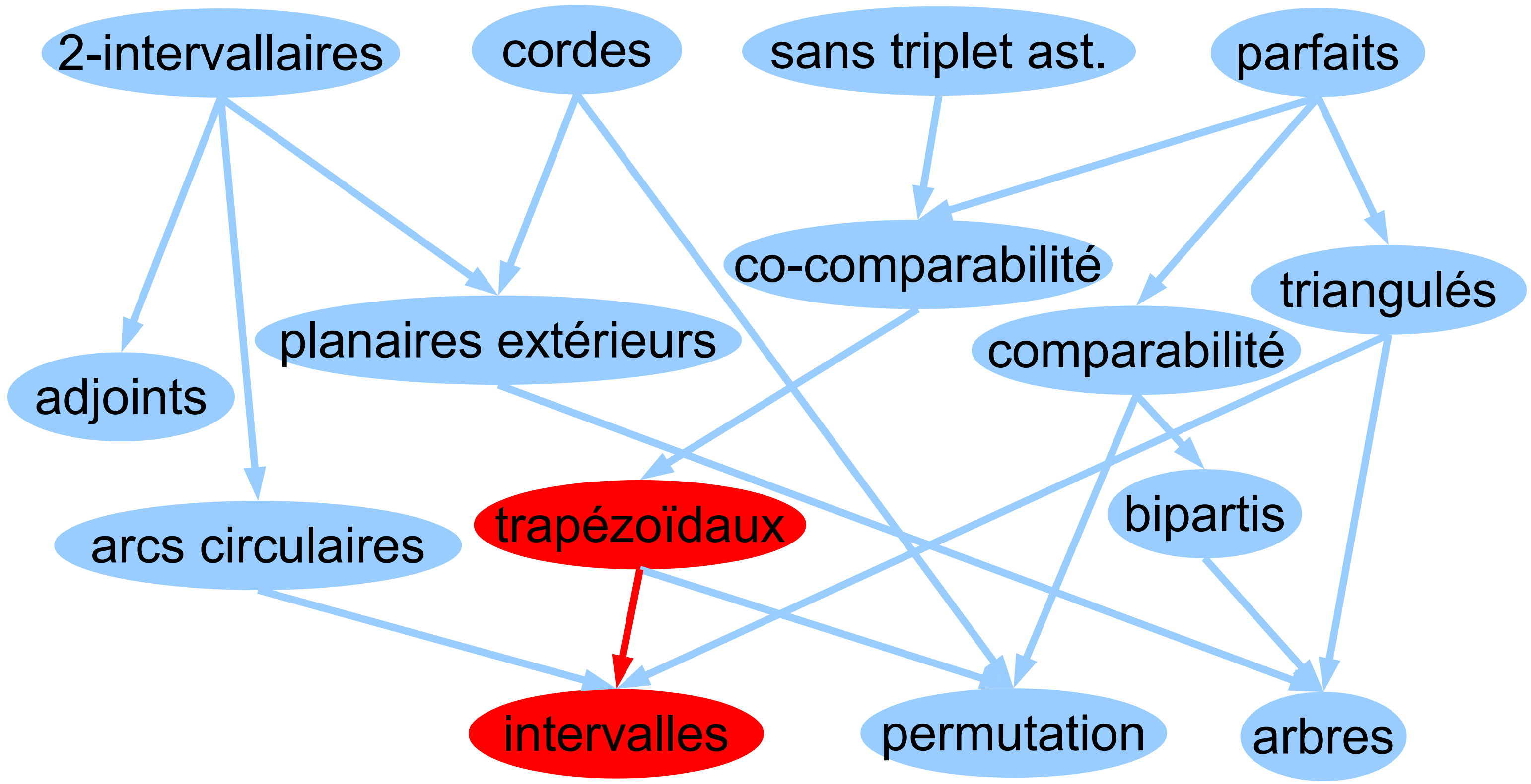
$\mathcal{I} = \{[0,2],[1,3],[3,4]\}$ $\Longrightarrow$ $\mathcal{T} = \{([0,2],[0,2]),([1,3],[1,3]),([3,4],[3,4])\}$





La **classe** de graphes d'intervalles **est incluse** dans celle des graphes trapézoïdaux.

Graphe d'inclusion des classes de graphes



Graphe d'inclusion des classes de graphes

Information System on Graph Classes and their Inclusions

Global	ISGCI home	Java	All classes	References	Smallgraphs		Find class
This class	Definition	Inclusions	Problems	[+]Details	[-]Hide details		

Graphclass: interval

Definition:
A graph is an interval graph if it has an intersection model consisting of intervals on a straight line.

References: [453]

Equivalent classes: [+]Details

- 1-DIR
- AT-free ∩ chordal
- C_4-free ∩ co-comparability
- $(C_{n+4},T_2,X_{31},XF_2^{n+1},XF_3^{n})$-free
- boxicity 1
- chordal ∩ co-comparability

Complement classes:

- $(\overline{C_{n+4}},\overline{T_2},\overline{X_{31}},\text{co-}XF_2^{n+1},\text{co-}XF_3^{n})$-free
- co-chordal ∩ comparability
- co-chordal ∩ superperfect
- co-interval
- comparability graphs of semiorders

Related classes:

- 2-interval
- 2-thin
- (P_5,bull)-free ∩ interval
- bounded tolerance
- circular arc
- claw-free ∩ interval
- clique graphs of interval

Inclusions

Minimal superclasses: [+]Details

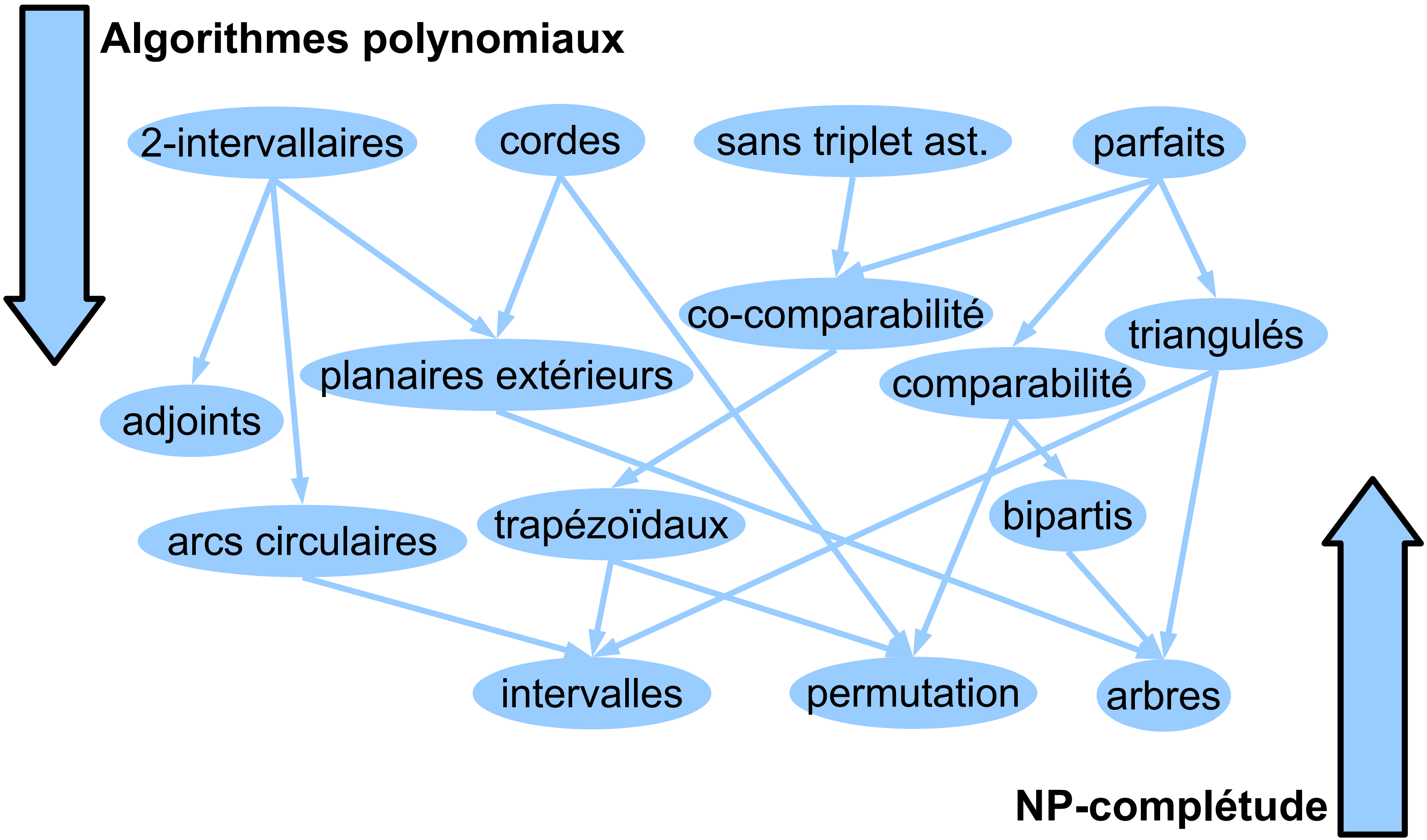
- 2-DIR
- 2-thin
- (C_{n+4},S_3,net)-free
- (C_{n+4},T_2,XF_2^{n+1})-free

Maximal subclasses: [+]Details

- $(2K_2,C_4,C_5,S_3,\text{net},\text{rising sun})$-free
- $(C_{n+4},P_5,\text{bull})$-free
- $(C_{n+4},S_3,\text{claw},\text{net})$-free
- (P_5,bull)-free ∩ interval

http://www.graphclasses.org/
by H.N. de Ridder et al. 2001-2012

Graphe d'inclusion des classes de graphes



Graphe d'inclusion des classes de graphes

Information System on Graph Classes and their Inclusions

Global	ISGCI home	Java	All classes	References	Smallgraphs		Find class
This class	Definition	Inclusions	Problems	[+]Details	[-]Hide details		

Graphclass: interval

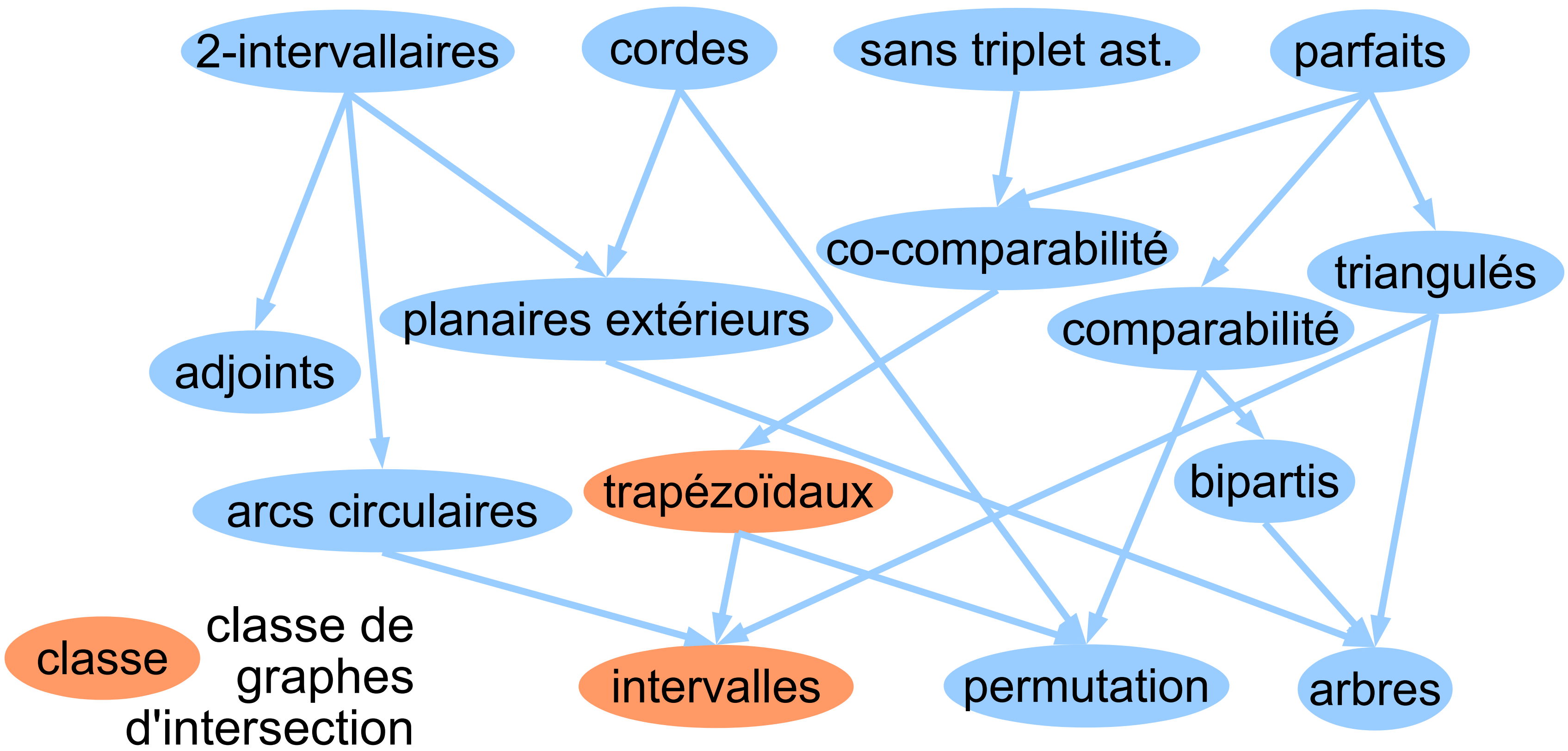
Definition:
A graph is an interval graph if it has an intersection model consisting of intervals on a straight line.

Problems

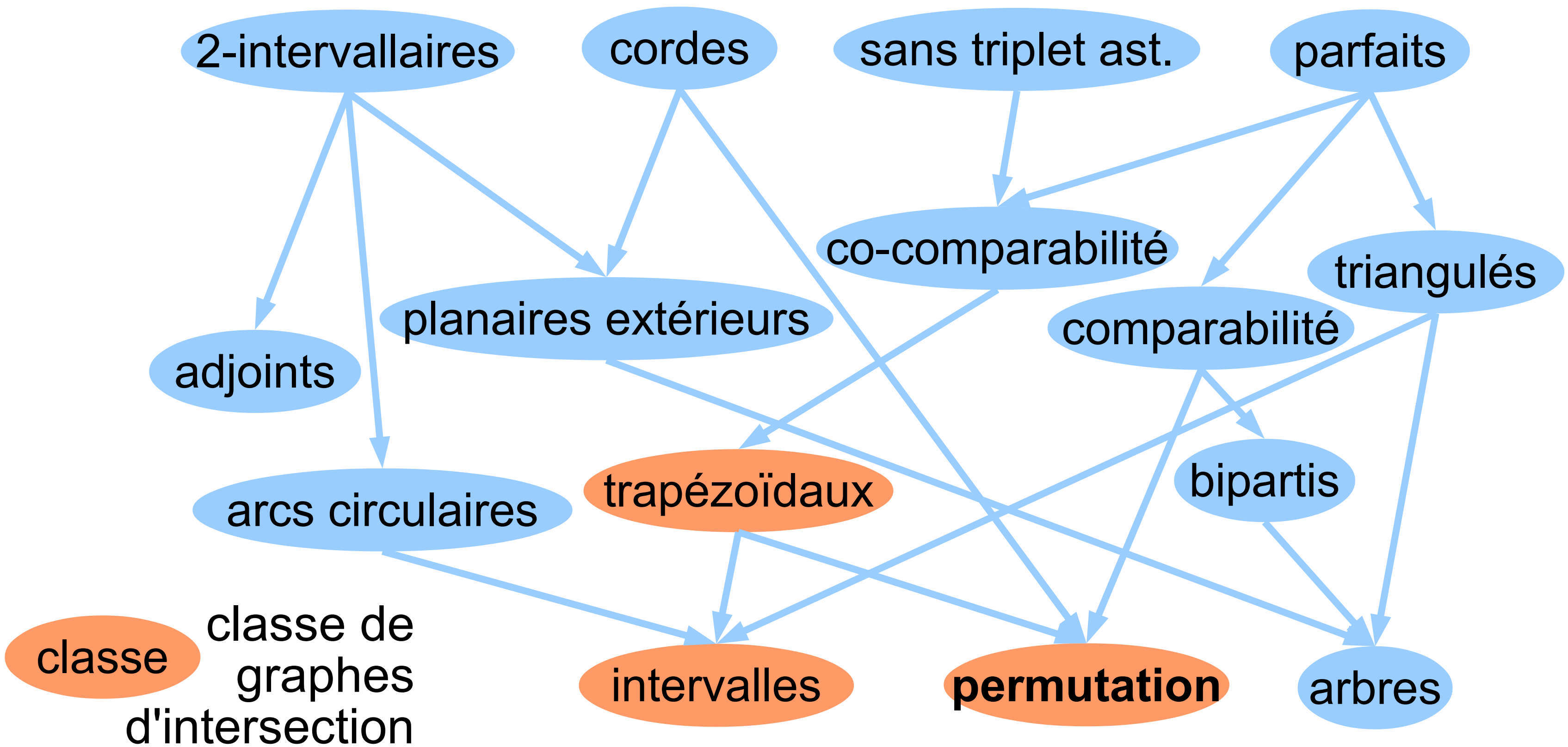
3-Colourability [?]	Linear	[+]Details
Clique [?]	Polynomial	[+]Details
Clique cover [?]	Linear	[+]Details
Cliquewidth [?]	Unbounded	[+]Details
Cliquewidth expression [?]	Unbounded or NP-complete	[+]Details
Colourability [?]	Linear	[+]Details
Domination [?]	Linear	[+]Details
Independent set [?]	Linear	[+]Details
Recognition [?]	Linear	[+]Details
Treewidth [?]	Polynomial	[+]Details
Weighted clique [?]	Polynomial	[+]Details
Weighted independent set [?]	Linear	[+]Details

http://www.graphclasses.org/
by H.N. de Ridder et al. 2001-2012

Graphe d'inclusion des classes de graphes

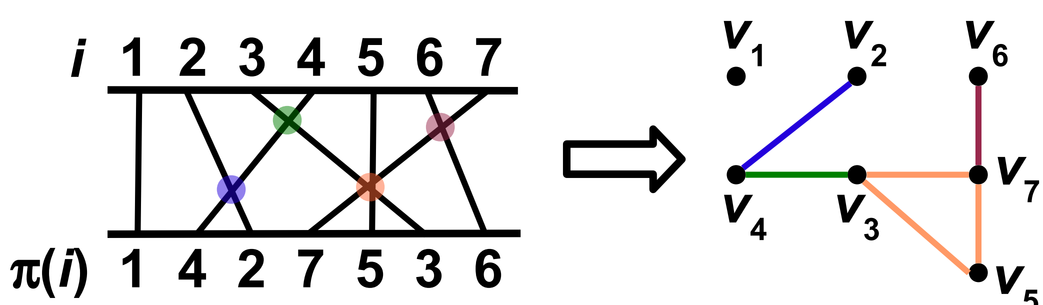


Graphe d'inclusion des classes de graphes

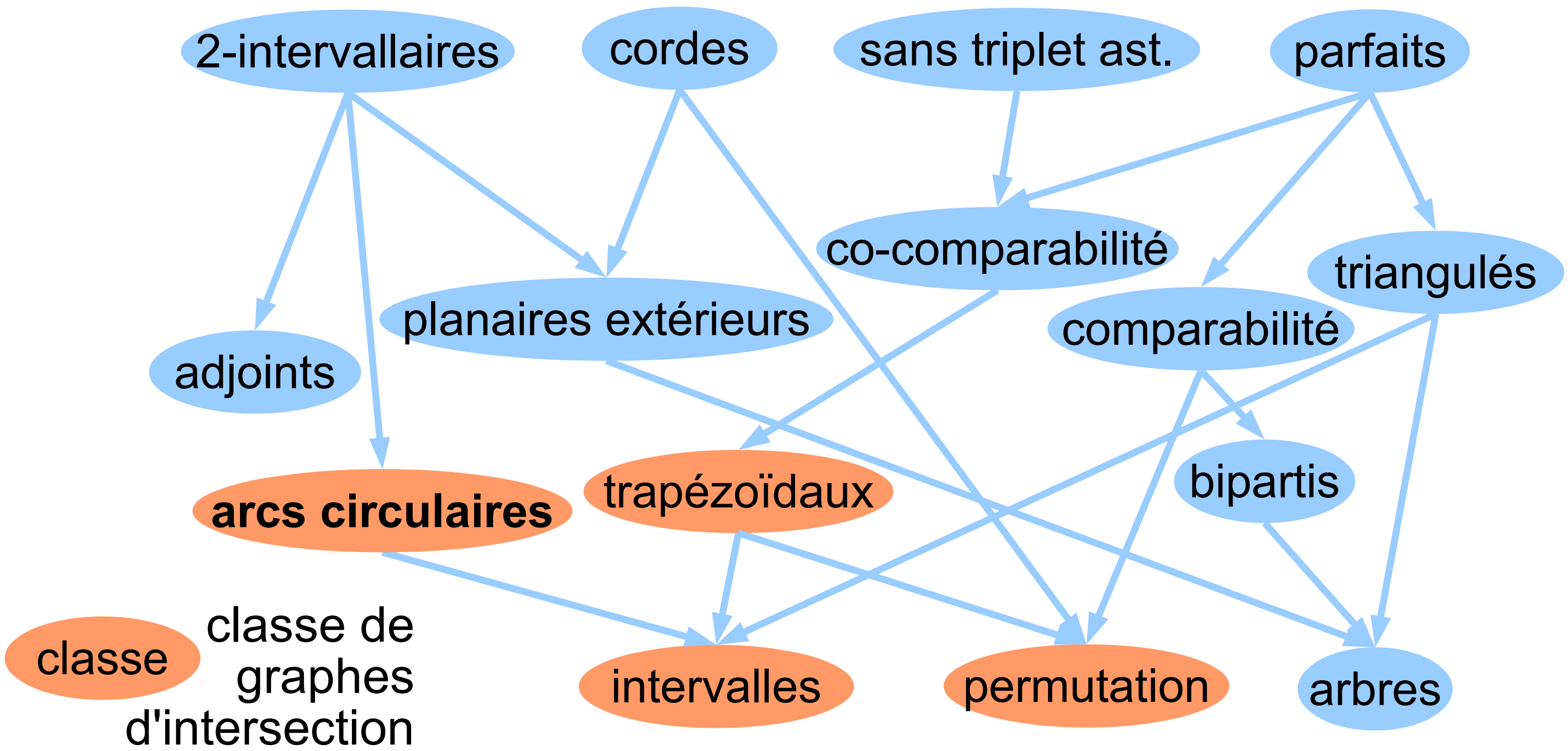


Graphe de **permutation** : graphe d'intersection des **segments (*k*,*k*)**.

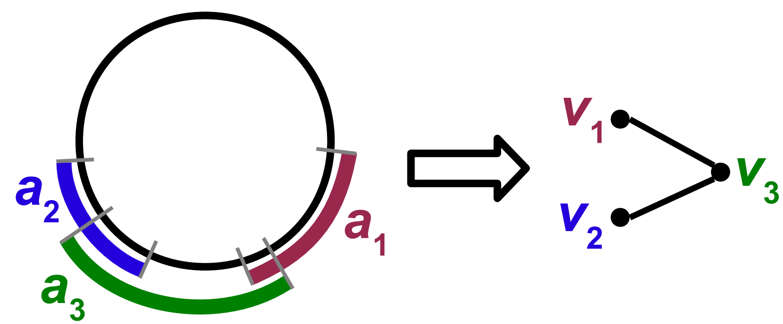
ligne des *i*   ligne des π(*i*)



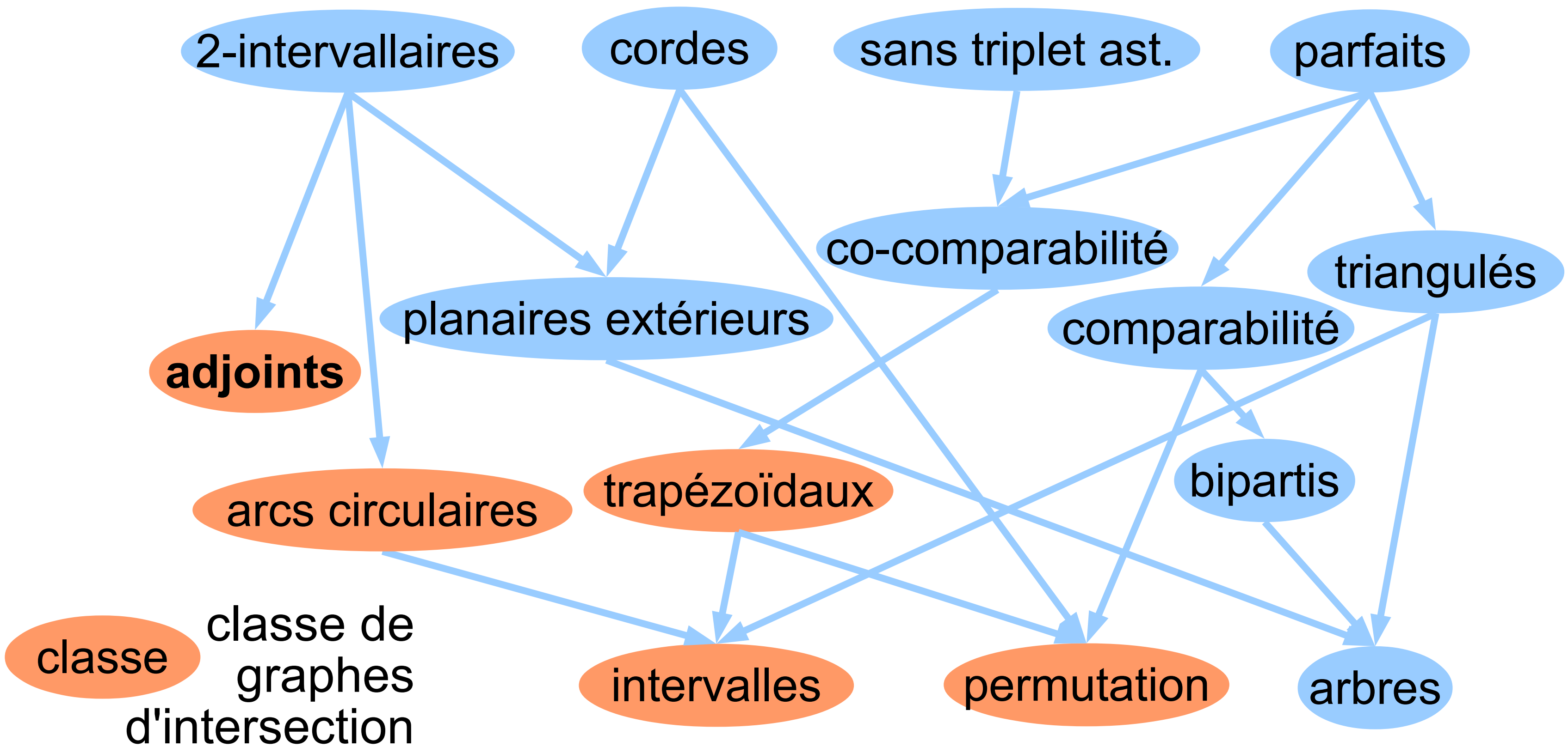
Graphe d'inclusion des classes de graphes



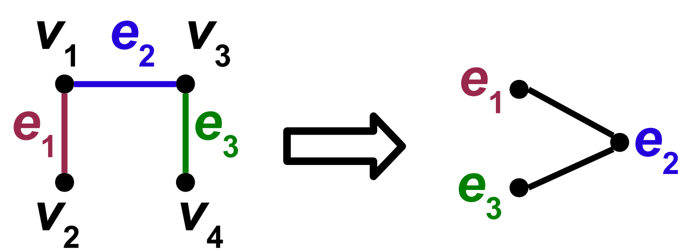
Graphe d'**arcs circulaires** : graphe d'intersection d'**arcs d'un cercle**.



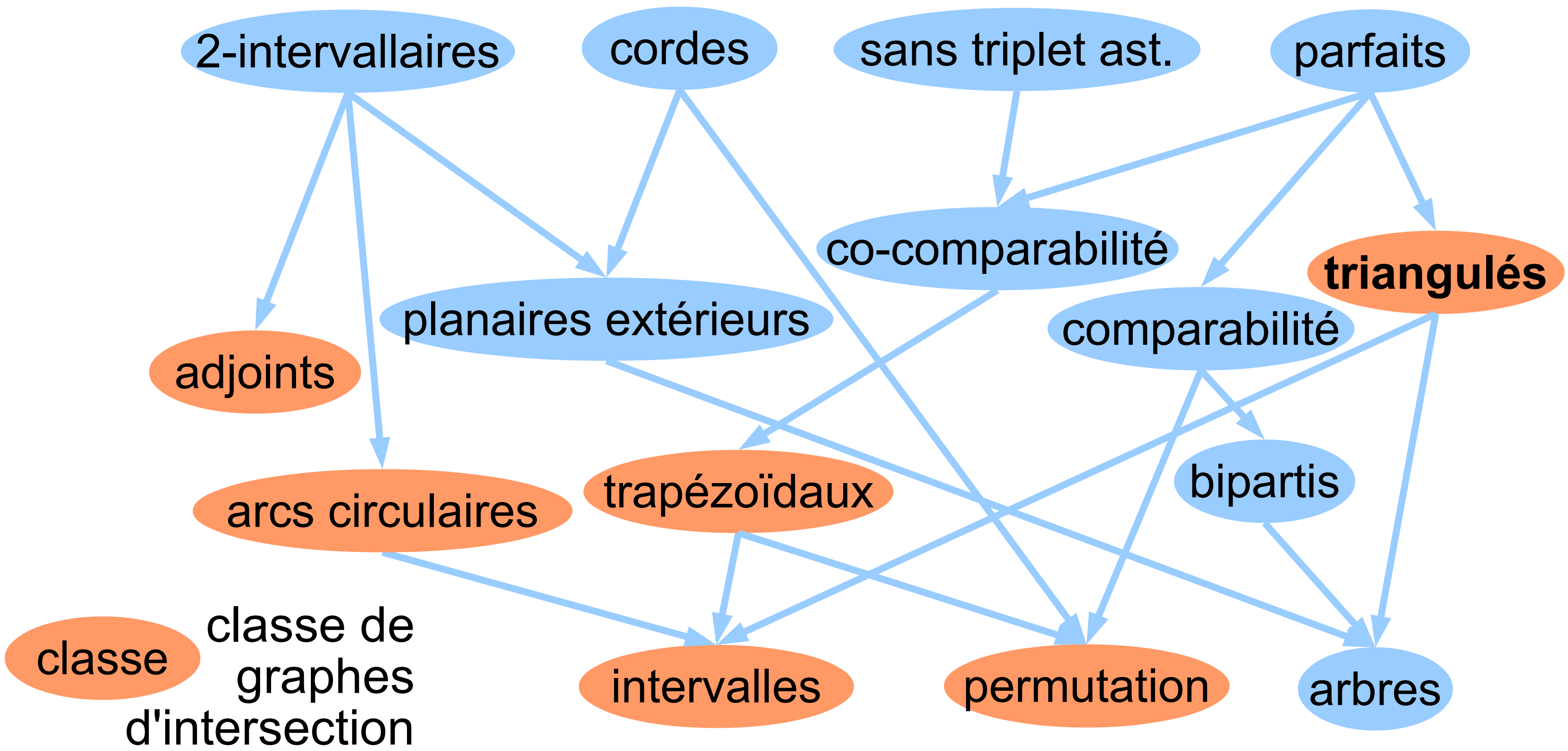
Graphe d'inclusion des classes de graphes



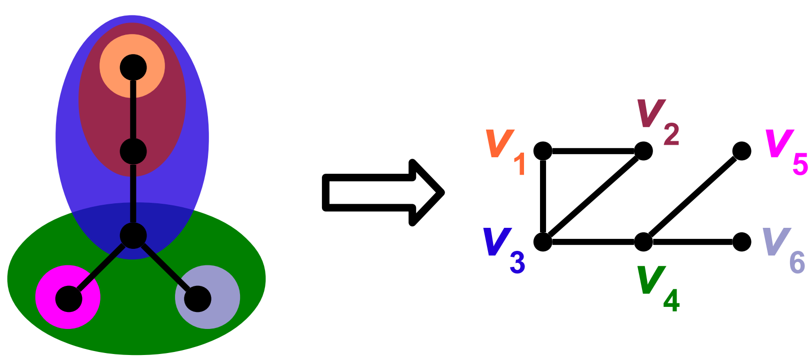
Graphe **adjoint** (***line graph***) : graphe d'intersection des **arêtes d'un graphe**.



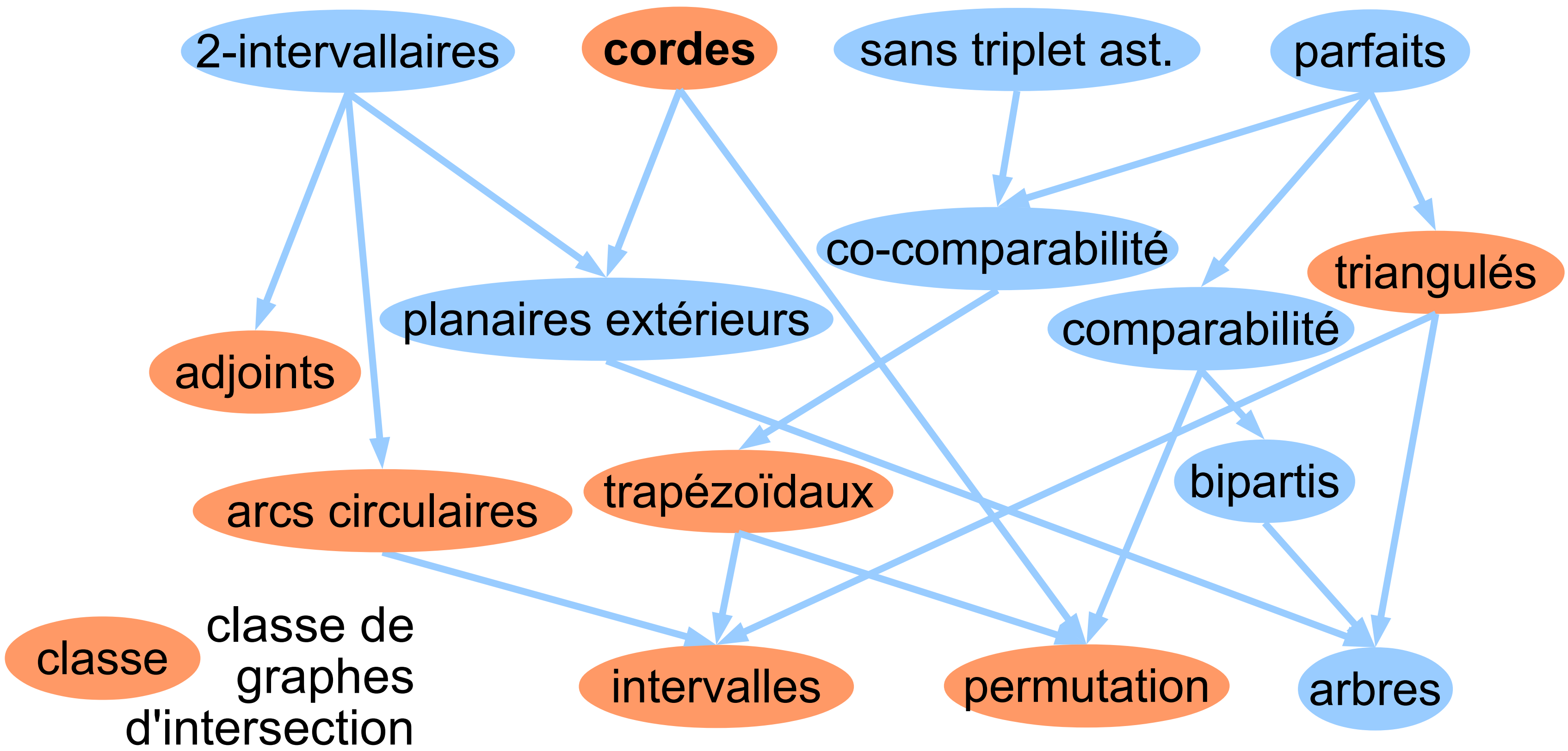
Graphe d'inclusion des classes de graphes



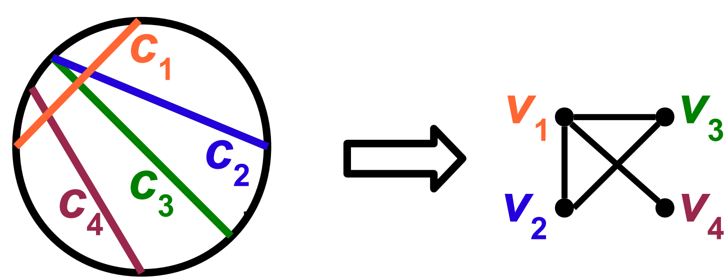
Graphe **triangulé** (***chordal***) : graphe d'intersection d'une famille de **sous-arbres d'un arbre**.



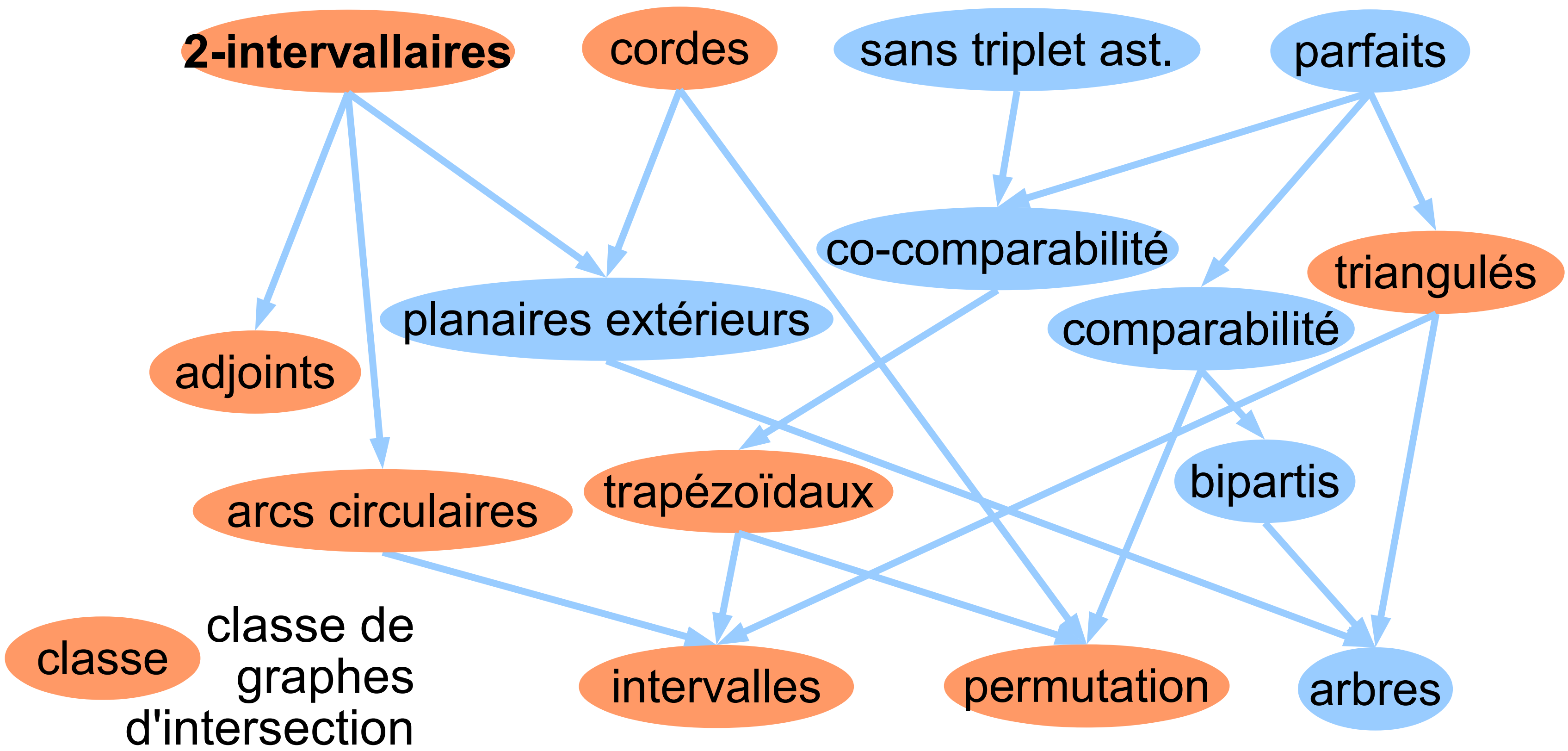
Graphe d'inclusion des classes de graphes



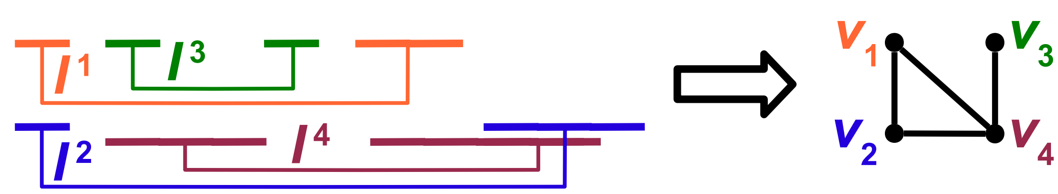
Graphe **de cordes** (***circle***) : graphe d'intersection des **cordes d'un cercle**.



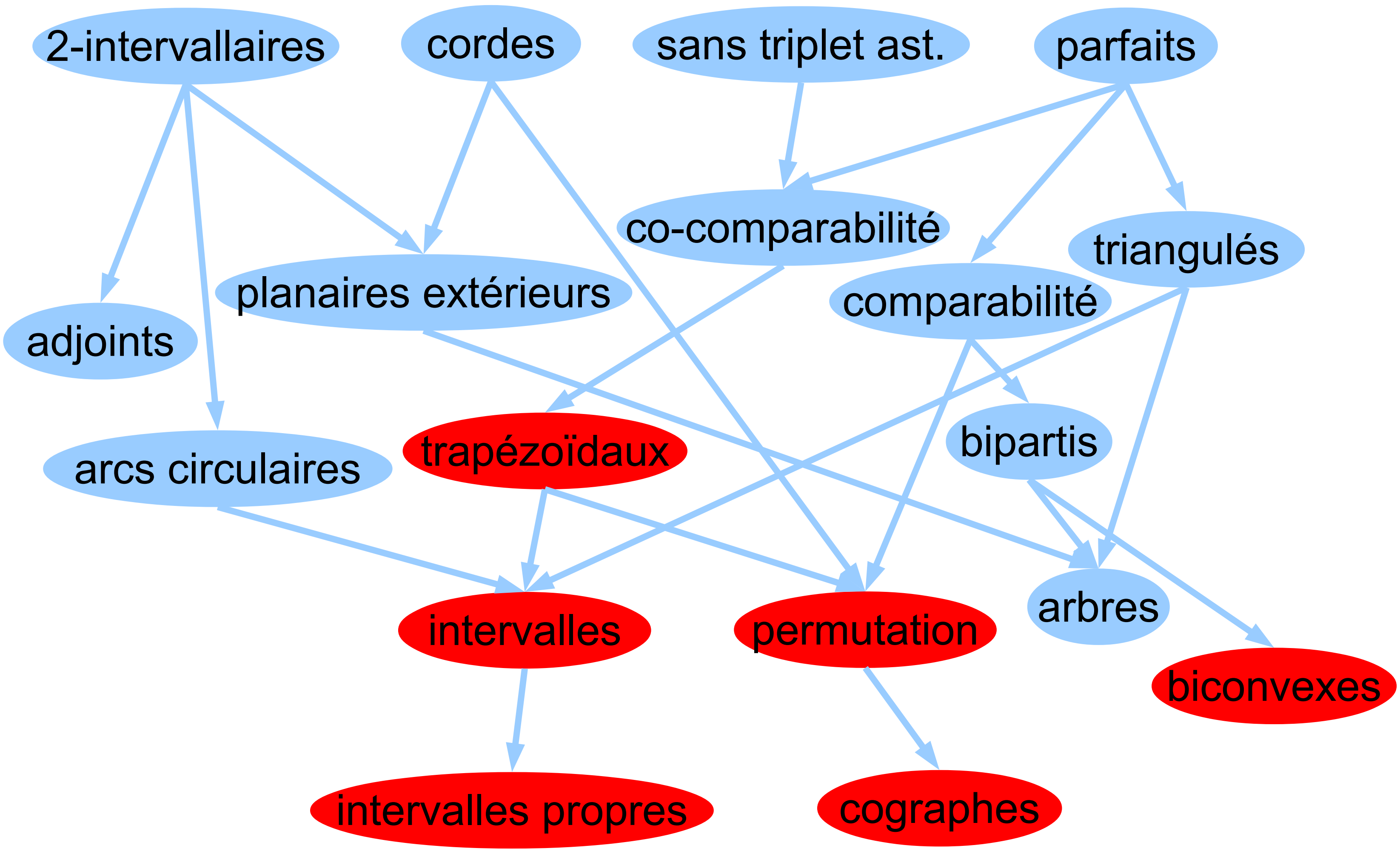
Graphe d'inclusion des classes de graphes



Graphe **2-intervallaire** :
graphe d'intersection
d'unions de deux intervalles.



Les classes de cette présentation



graphes de permutation
avec réalisation
sans motif 3,1,4,2 ni 2,4,1,3

Plan

- Graphes d'intersection
- Algorithme de parcours en largeur avec priorités
- Codage des voisinages
- Contiguïté et linéarité

Parcours en largeur des graphes d'intervalles

Parcours en largeur (BFS) :

- Lister les les voisins du premier sommet non visité
- Les ajouter à la fin de la file des non visités

Parcours en largeur des graphes d'intervalles

Parcours en largeur (BFS) suivant un ordre de priorité σ :

- Lister les les voisins du premier sommet non visité
- Les ajouter à la fin de la file des non visités
  dans l'ordre donné par σ

→ “Arbre BFS” :
chaque sommet est fils de son voisin responsable de l'ajout dans la file.

→ Complexité : $O(n+m)$

Parcours en largeur des graphes d'intervalles

Parcours en largeur (BFS) suivant un ordre de priorité σ :
- Lister les les voisins du premier sommet non visité
- Les ajouter à la fin de la file des non visités
  dans l'ordre donné par σ

→ “Arbre BFS” :
chaque sommet est fils de son voisin responsable de l'ajout dans la file.

→ Complexité : $O(n+m)$

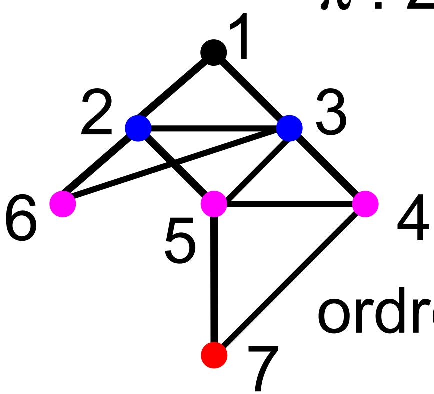
Algorithme en $O(n)$ pour les graphes d'intervalles,
les graphes de permutation, les graphes trapézoïdaux,
étant donné une réalisation

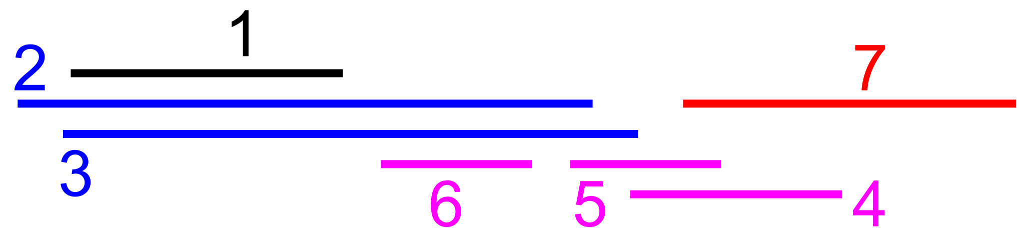
Gambette & Crespelle, IWOCA 2009 ; IPL, 2010

Parcours en largeur des graphes d'intervalles

Entrée : ordre des bornes :

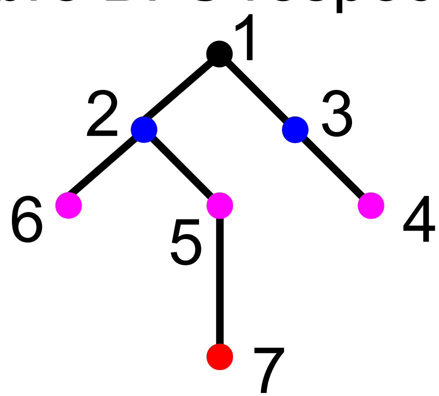
π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





ordre de priorité σ : 1 2 3 4 5 6 7

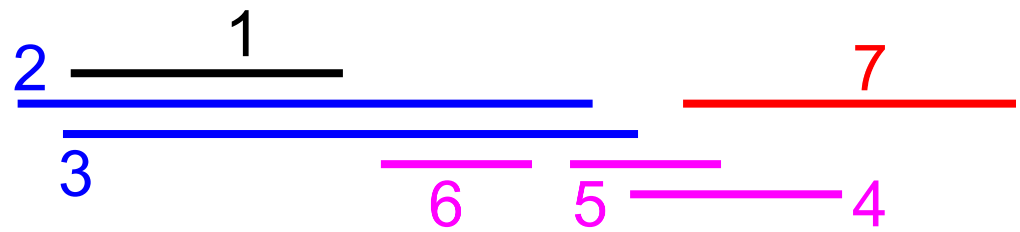
Sortie : arbre BFS respectant σ

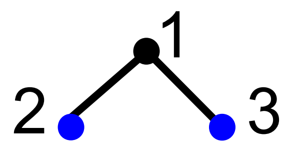


Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d

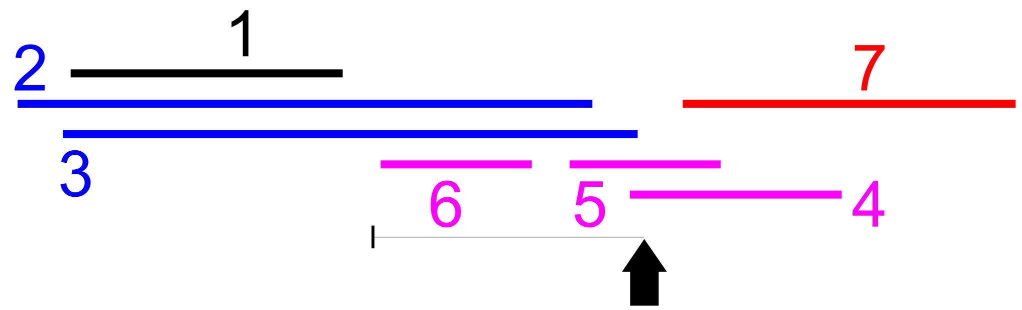


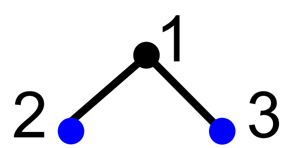


Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 3

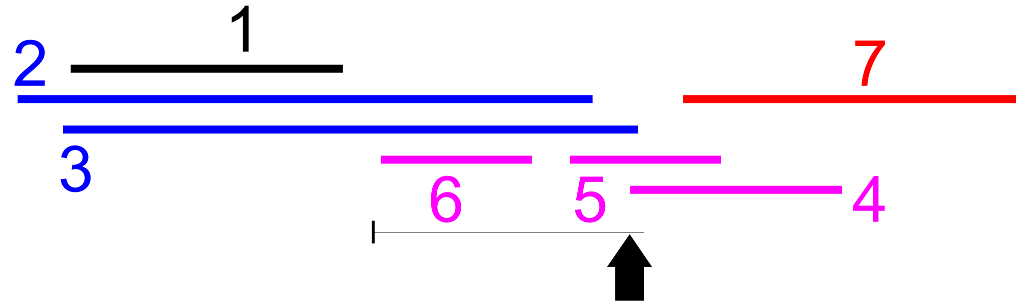
borne droite : 3

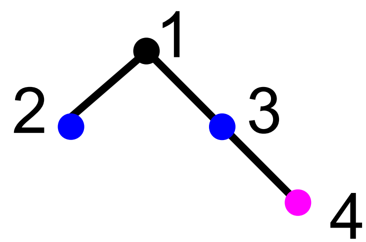
priorité' : 1

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 3

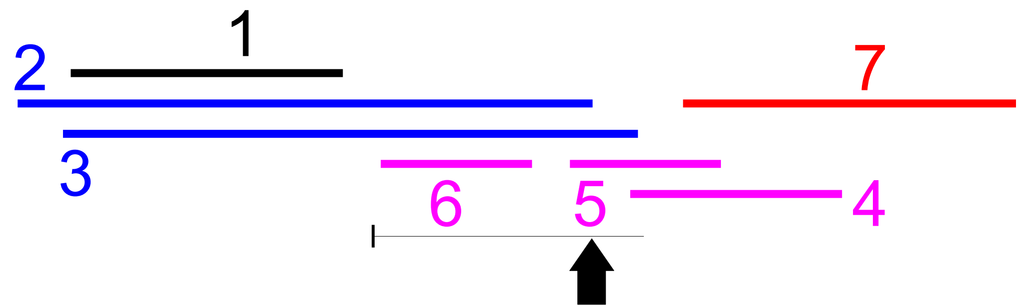
borne droite : 4

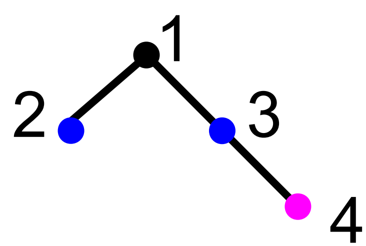
priorité' : 1 3

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 2

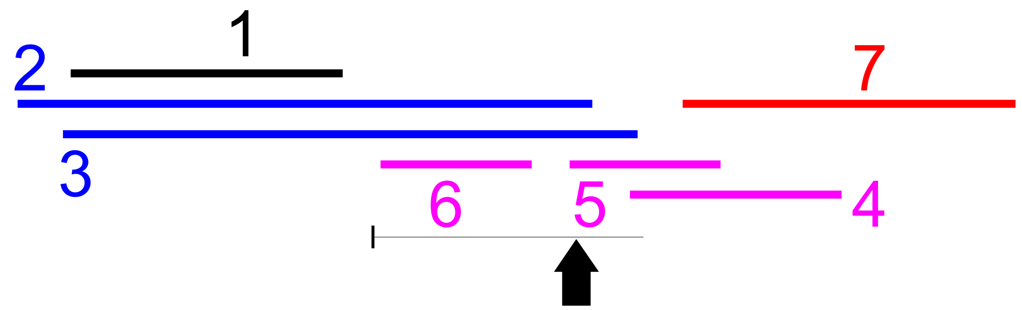
borne droite : 4

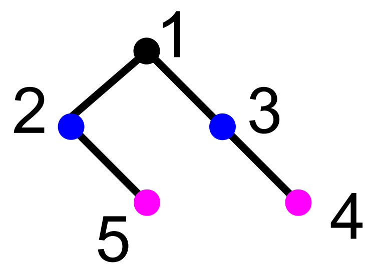
priorité' : 1 2 3

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 2

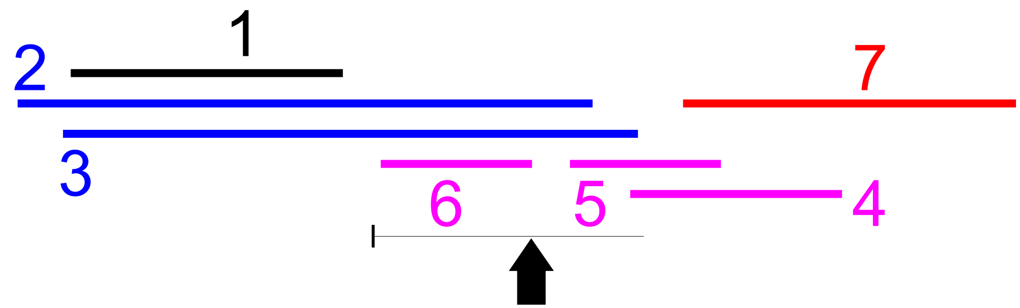
borne droite : 4

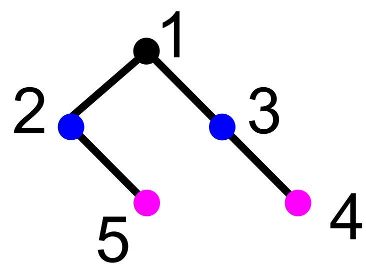
priorité' : 1 2 3

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 2

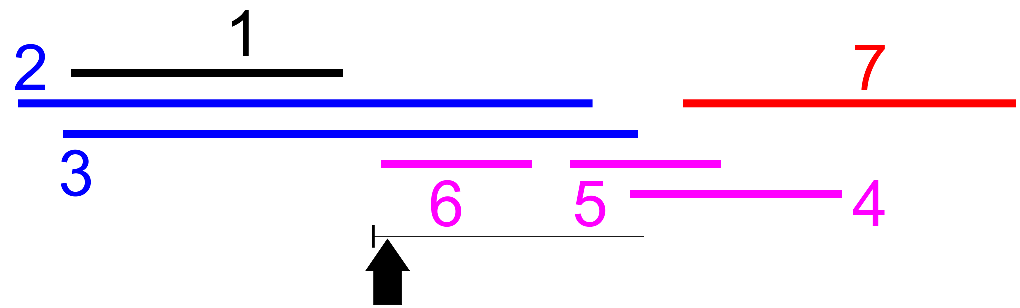
borne droite : 4

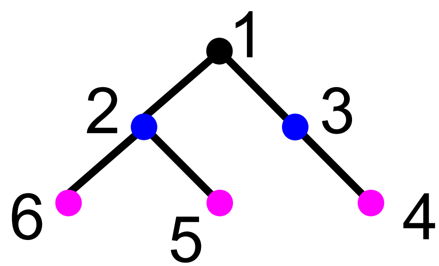
priorité' : 1 2 3

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 2

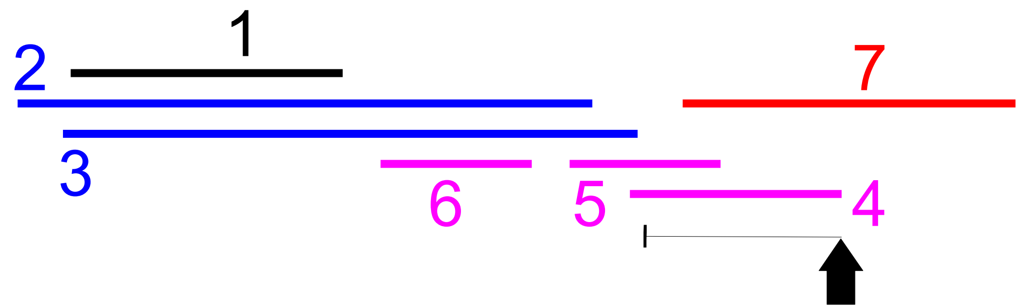
borne droite : 4

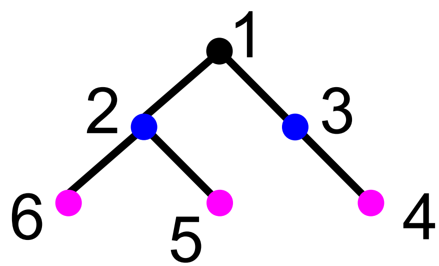
priorité' : 1 2 3

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 4

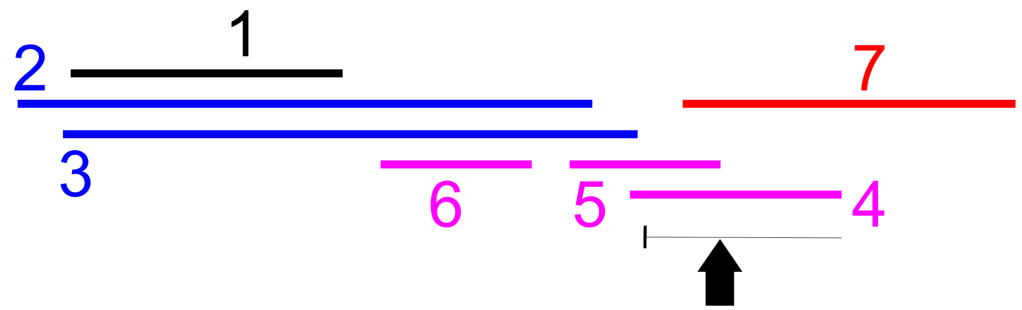
borne droite : 4

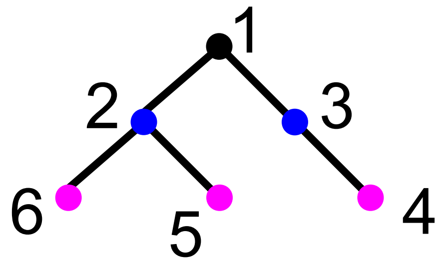
priorité' : 1 2 3 4

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 5

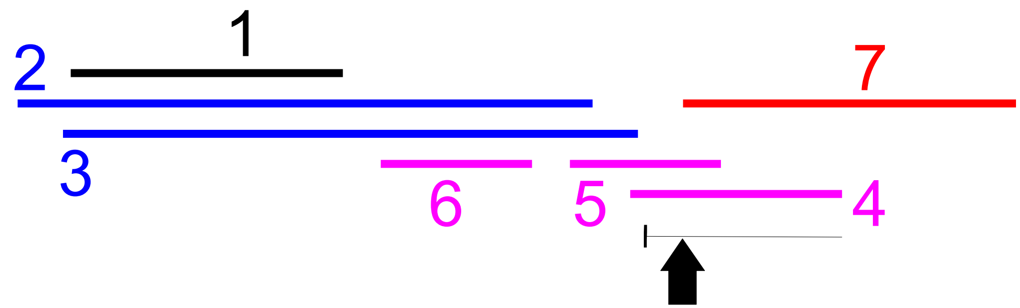
borne droite : 4

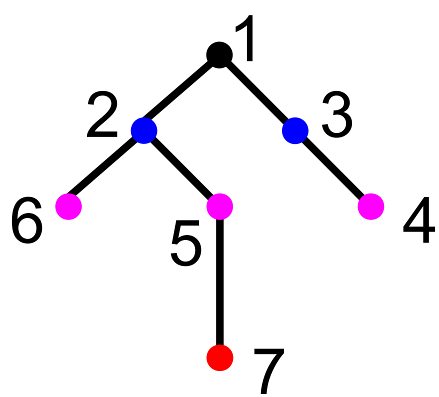
priorité' : 1 2 3 5 4

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d





père courant : 5

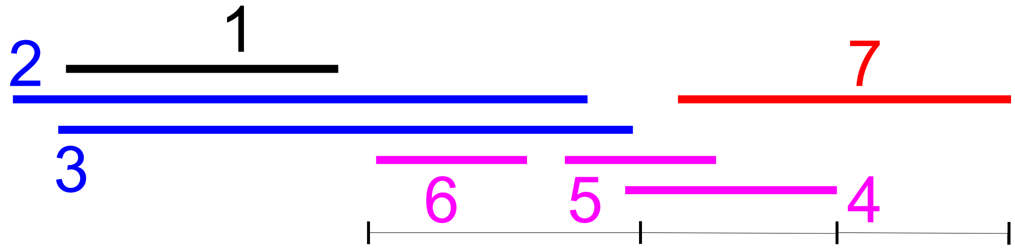
borne droite : 7

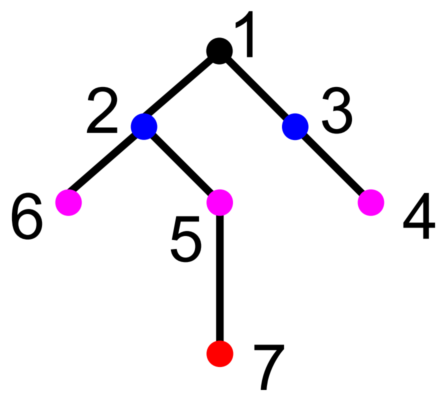
priorité' : 1 2 3 5 4

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d



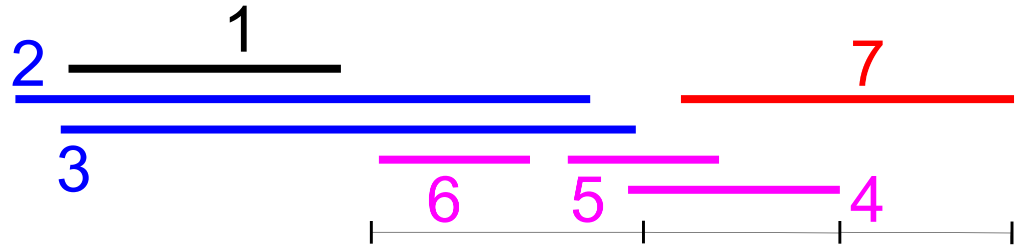


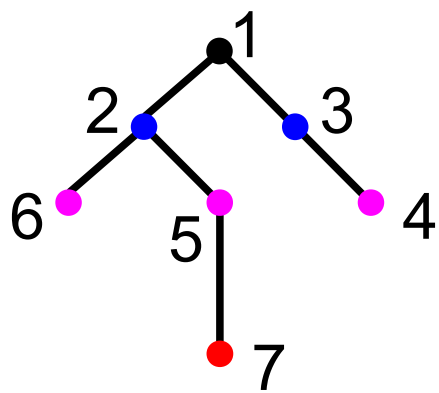
+ même chose vers la gauche, suivie de la combinaison des listes de priorité'

Parcours en largeur des graphes d'intervalles

σ : 1 2 3 4 5 6 7

π : 2g 3g 1g 1d 6g 6d 5g 2d 4g 3d 7g 5d 4d 7d



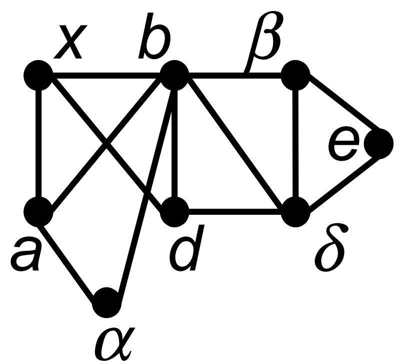


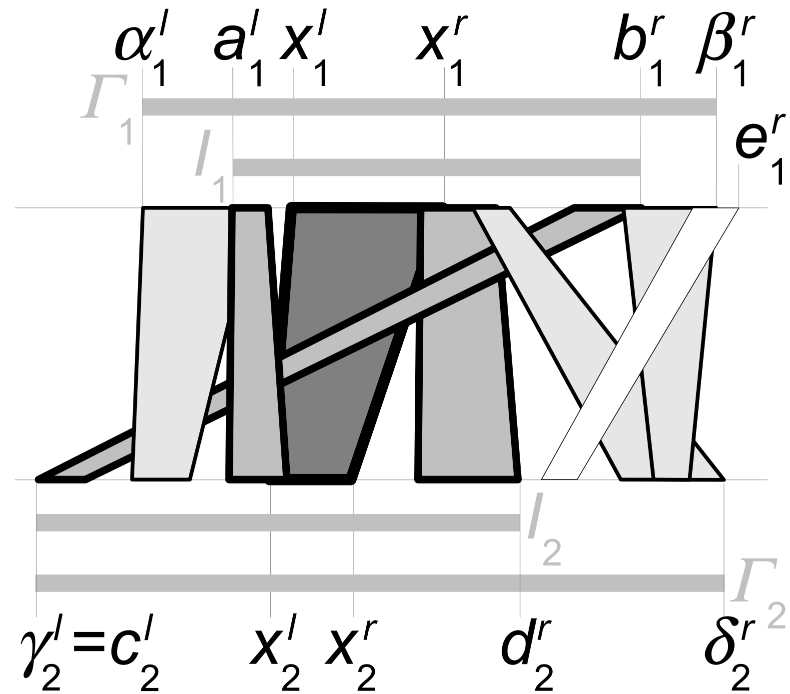
chaque borne visitée une seule fois : algorithme en O(*n*)

Parcours en largeur des graphes trapézoïdaux

Extension de l'algorithme :

- Exploration de π_1 et π_2 vers la gauche et vers la droite
- Gestion plus complexe de la liste des parents visités





Gambette & Crespelle, IPL, 2010

Plan

- Graphes d'intersection
- Algorithme de parcours en largeur avec priorités
- Codage des voisinages
- Contiguïté et linéarité

Contexte

Algorithmes sur les grands graphes

- Stockage du graphe en mémoire — Encodage compact
- Requêtes sur le graphe — Temps de réponse réduit

Contexte

Algorithmes sur les grands graphes

- Stockage du graphe en mémoire — Encodage compact
- Requêtes sur le graphe — Temps de réponse réduit

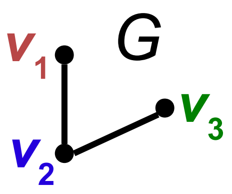
Bon encodage vis à vis des requêtes d'adjacence :
Nombreuses classes de graphes d'intersection

Contexte

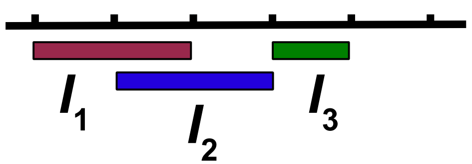
Algorithmes sur les grands graphes

- Stockage du graphe en mémoire — Encodage compact
- Requêtes sur le graphe — Temps de réponse réduit

Bon encodage vis à vis des requêtes d'adjacence :
Nombreuses classes de graphes d'intersection



$\mathcal{I} = \{[0,2],[1,3],[3,4]\}$



1 ∈ [0,2] :
1 et 2 adjacents

Encodage en O(n)

Requêtes d'adjacence en O(1)

2 sommets adjacents ssi
une borne de l'intervalle de l'un
est incluse dans l'intervalle de
l'autre.

Contexte

Algorithmes sur les grands graphes

- Stockage du graphe en mémoire — Encodage compact
- Requêtes sur le graphe — Temps de réponse réduit

Bon encodage vis à vis des requêtes d'adjacence :
Nombreuses classes de graphes d'intersection

Bon encodage vis à vis des requêtes de voisinage ?

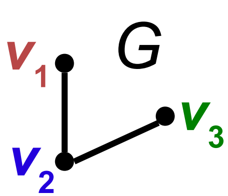
Contexte

Algorithmes sur les grands graphes

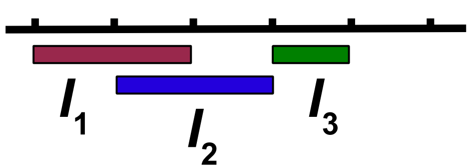
- Stockage du graphe en mémoire — Encodage compact
- Requêtes sur le graphe — Temps de réponse réduit

Bon encodage vis à vis des requêtes d'adjacence :
Nombreuses classes de graphes d'intersection

Bon encodage vis à vis des requêtes de voisinage ?
Exemple : graphes d'intervalles



$\mathcal{I} = \{[0,2],[1,3],[3,4]\}$



Encodage en O(n)

Requêtes de voisinage en O(n)

O(d) ?

Contexte

Algorithmes sur les grands graphes

- Stockage du graphe en mémoire — Encodage compact
- Requêtes sur le graphe — Temps de réponse réduit

Bon encodage vis à vis des requêtes d'adjacence :
Nombreuses classes de graphes d'intersection

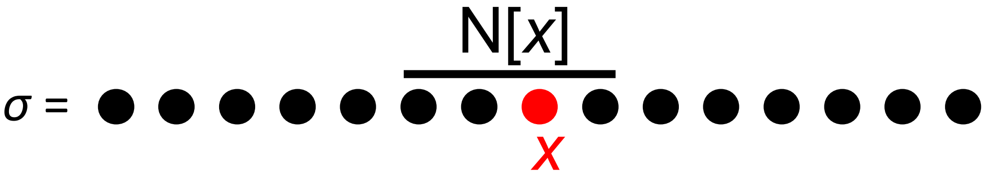
Bon encodage vis à vis des requêtes de voisinage ?
Encodage en **O(*n*)** avec requêtes de voisinage en **O(*d*)**
pour graphes d'intervalles et graphes de permutation.
→ extension aux graphes quelconques par complétion
→ exploitation algorithmique de la structure des voisinages

Une approche naturelle

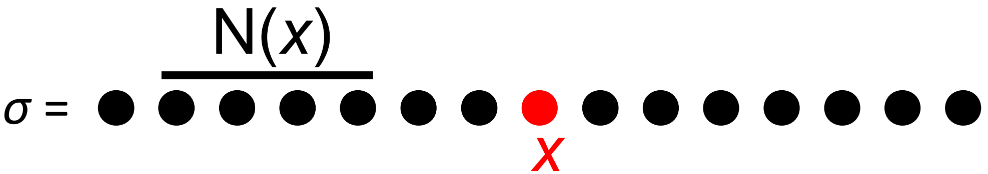
Trouver un ordre *σ* sur les sommets tel que :

le voisinage (ouvert N(*x*) ou fermé N[*x*]) de tout sommet *x* est un intervalle de *σ*

- Graphes d'intervalles propres



- Graphes biconvexes

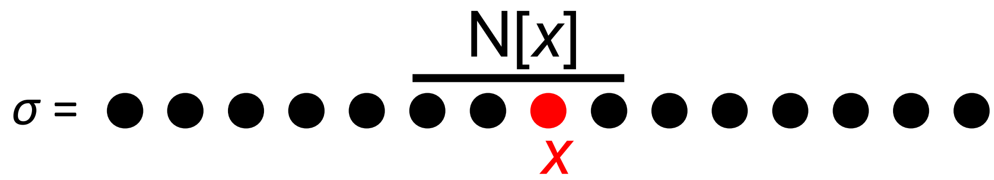


Une approche naturelle

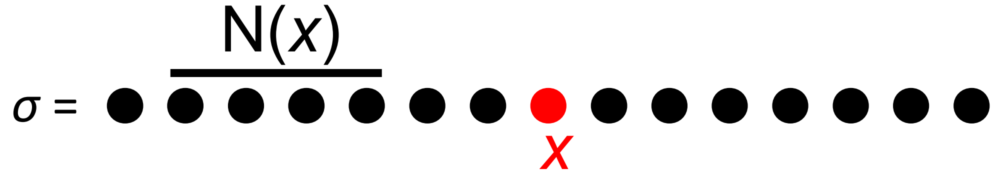
Trouver un ordre σ sur les sommets tel que :

le voisinage (ouvert N(x) ou fermé N[x]) de tout sommet x est un intervalle de σ

- Graphes d'intervalles propres



- Graphes biconvexes



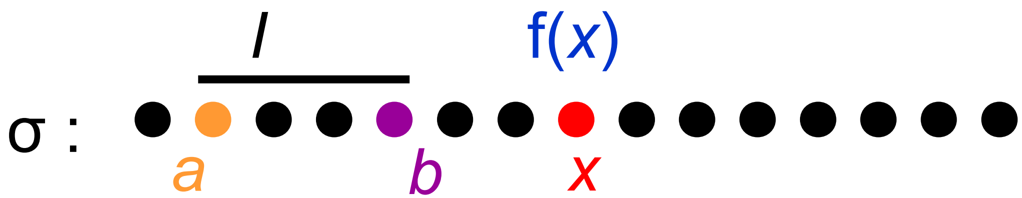
Généralisation, 2 paramètres :

- autoriser plusieurs intervalles : **contiguïté**
- autoriser plusieurs ordres : **linéarité**

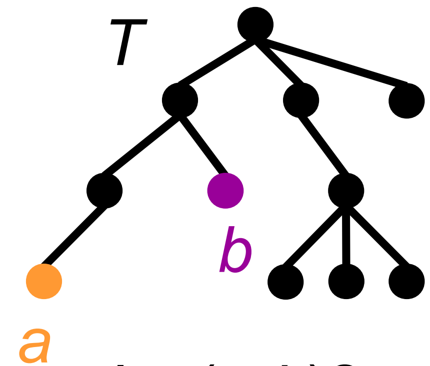
Voisinages dans les graphes de permutation

Arbres cartésiens augmentés :

Max d'une fonction entière sur un intervalle :



$$\max_{x \in I} f(x)?$$

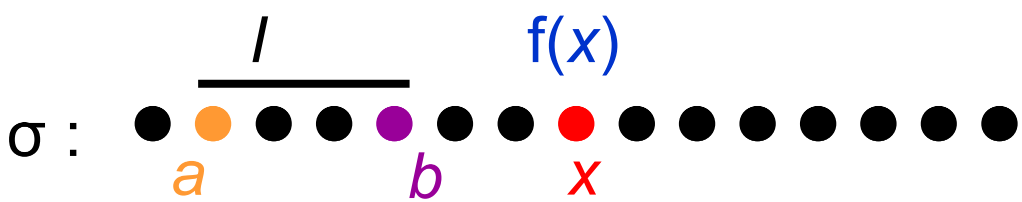


lca(a,b)?

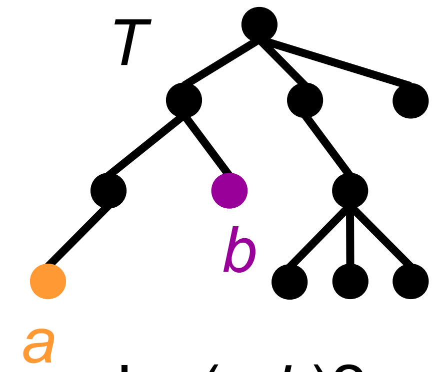
Voisinages dans les graphes de permutation

Arbres cartésiens augmentés :

Max d'une fonction entière sur un intervalle :



$$\max_{x \in I} f(x)?$$

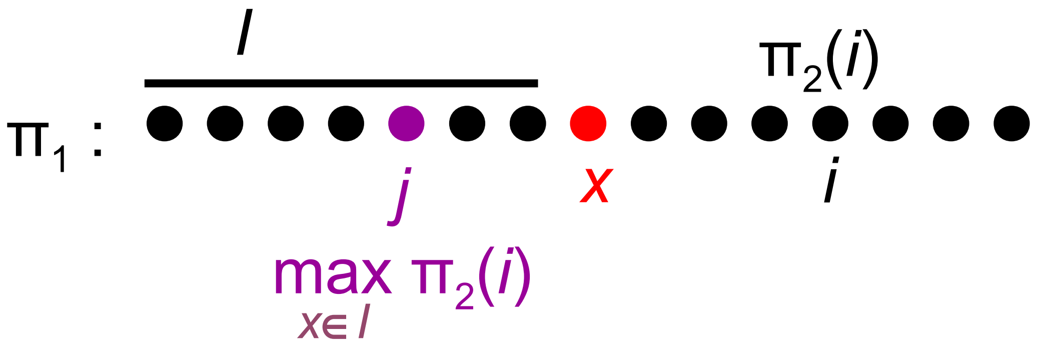


lca(a,b)?

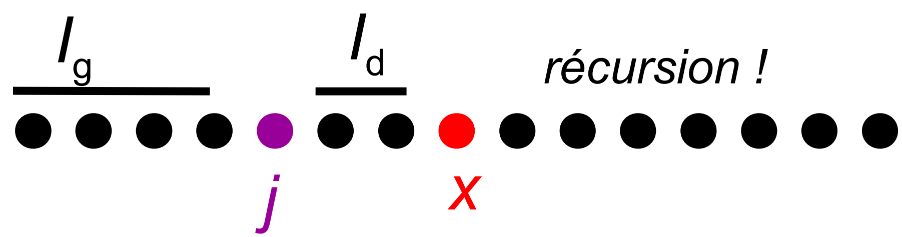
Précalcul de T en temps O(|T|)
pour répondre aux requêtes lca en O(1)

Harel & Tarjan 1984, Vuillemin 1980

Voisinages dans les graphes de permutation

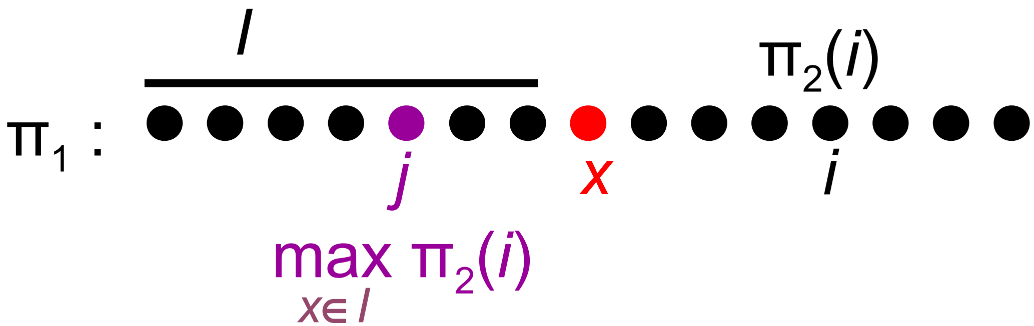


Si $\pi_2(j) > \pi_2(x)$ **alors** j est un voisin de x

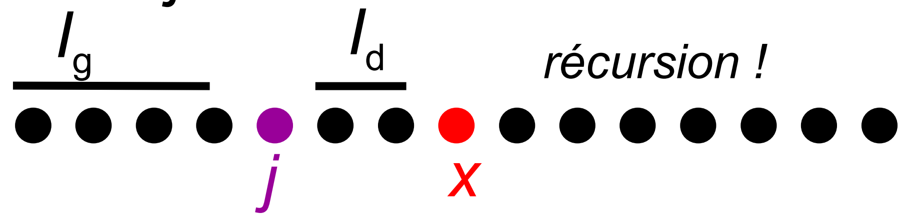


sinon x n'a pas de voisin dans I

Voisinages dans les graphes de permutation



Si $\pi_2(j) > \pi_2(x)$ **alors** j est un voisin de x



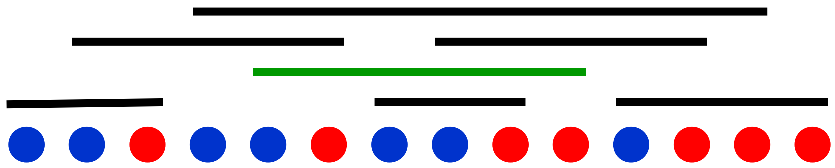
sinon x n'a pas de voisin dans I

Au plus deux appels récursif par un appel réussi
Chaque appel non réussi lancé par un appel réussi
→ complexité en temps en $O(d)$

Voisinages dans les graphes d'intervalles

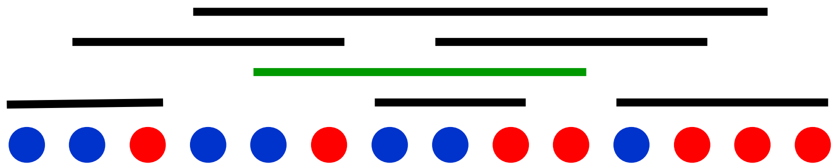
2 types de voisins :

- recouvrement
- inclusion



Voisinages dans les graphes d'intervalles

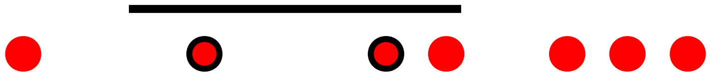
2 types de voisins :
- recouvrement
- inclusion



Recouvrements :

Bornes gauches

Bornes droites



Inclusions : graphe de permutation

Plan

- Graphes d'intersection
- Algorithme de parcours en largeur avec priorités
- Codage des voisinages
- Contiguïté et linéarité

Contiguïté et linéarité

- **Contiguïté** (ouverte) *k* : il existe un ordre des sommets tel que le voisinage ouvert de chaque sommet de *G* est une union de *k* intervalles dans l'ordre.

- **Linéarité** (ouverte) *k* : il existe *k* ordres des sommets tels que le voisinage ouvert de chaque sommet de *G* est constitué par une union de *k* intervalles, chaque intervalle provenant d'un des ordres

Contiguïté et linéarité

- **Contiguïté** (ouverte) k : il existe un ordre des sommets tel que le voisinage ouvert de chaque sommet de G est une union de k intervalles dans l'ordre.

- **Linéarité** (ouverte) k : il existe k ordres des sommets tels que le voisinage ouvert de chaque sommet de G est constitué par une union de k intervalles, chaque intervalle provenant d'un des ordres

Contiguïté fermée au plus k, pour k>1 : **NP-complet**

Goldberg, Golumbic, Kaplan & Shamir, JCB, 1995
Wang, Lau & Zhao, DAM, 2007

Bornes mutuelles entre ces paramètres :

cc(G)≥cl(G), oc(G)≥ol(G)≥cl(G)-1

Crespelle & Gambette, IWOCA 2009

Contiguïté et linéarité des cographes

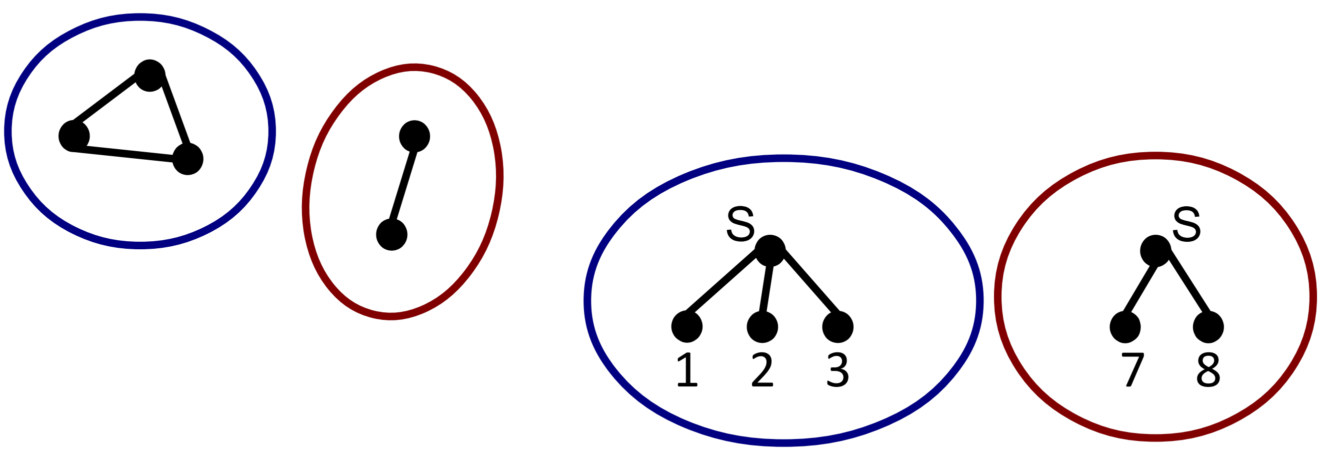
Borne sur les cographes : linéarité et contiguïté en O(log *n*)

Crespelle & Gambette, 2012

Définitions équivalentes des cographes :

- Graphes des permutations sans motif 3,1,4,2 ni 2,4,1,3
- Graphes sans P4 induit 
- Graphes obtenus par composition série (S) et parallèle (P)

cographe *G* coarbre(s) *T*



Contiguïté et linéarité des cographes

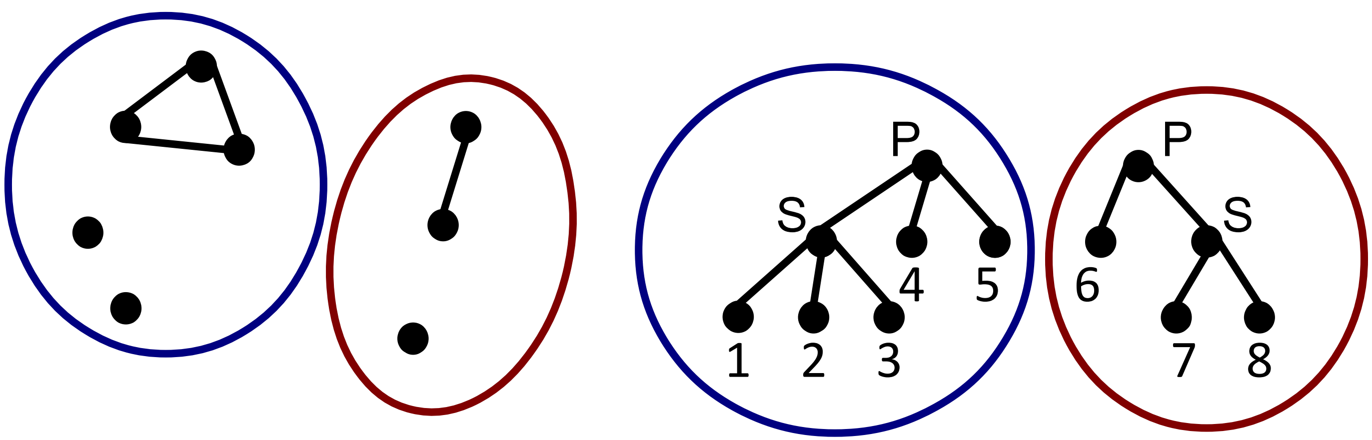
Borne sur les cographes : linéarité et contiguïté en O(log *n*)

Crespelle & Gambette, 2012

Définitions équivalentes des cographes :

- Graphes des permutations sans motif 3,1,4,2 ni 2,4,1,3
- Graphes sans P4 induit 
- Graphes obtenus par composition série (S) et parallèle (P)

cographe *G* coarbre(s) *T*



Contiguïté et linéarité des cographes

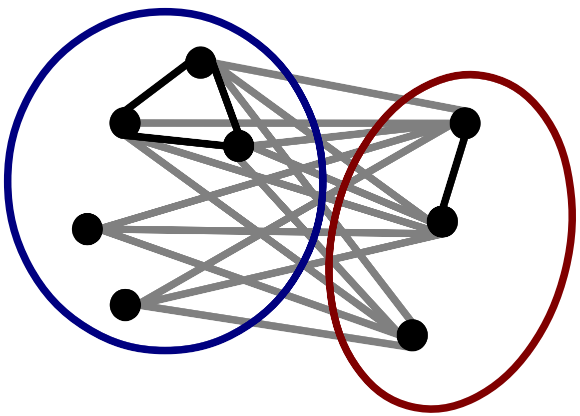
Borne sur les cographes : linéarité et contiguïté en O(log *n*)

Crespelle & Gambette, 2012

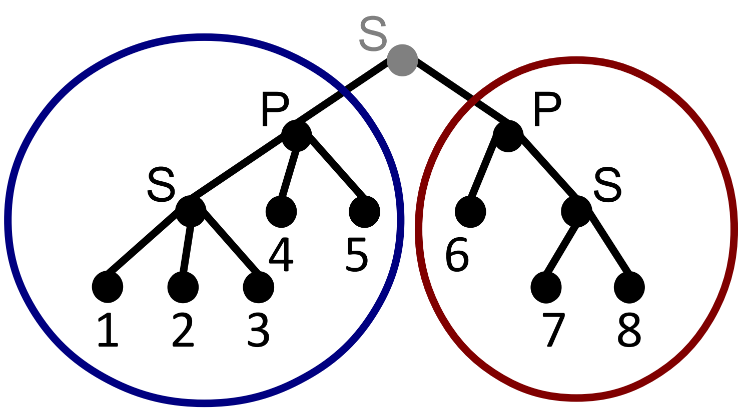
Définitions équivalentes des cographes :

- Graphes des permutations sans motif 3,1,4,2 ni 2,4,1,3
- Graphes sans P4 induit 
- Graphes obtenus par composition série (S) et parallèle (P)

cographe *G*



coarbre *T*



Contiguïté et linéarité des cographes

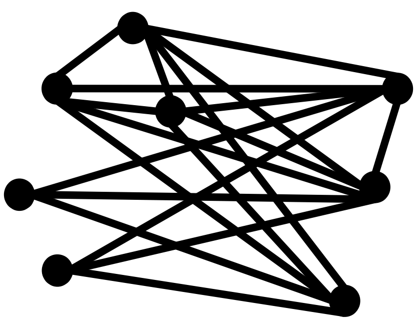
Borne sur les cographes : linéarité et contiguïté en O(log *n*)

Crespelle & Gambette, 2012

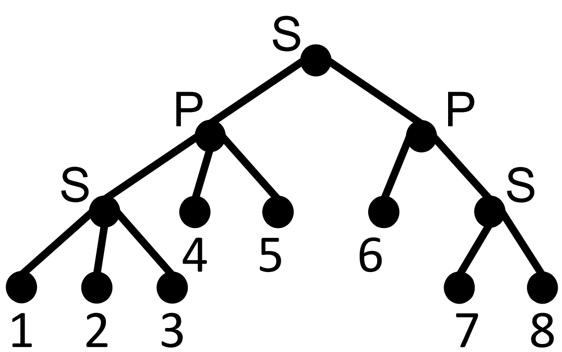
Définitions équivalentes des cographes :

- Graphes des permutations sans motif 3,1,4,2 ni 2,4,1,3
- Graphes sans P4 induit 
- Graphes obtenus par composition série (S) et parallèle (P)

cographe *G*



coarbre *T*



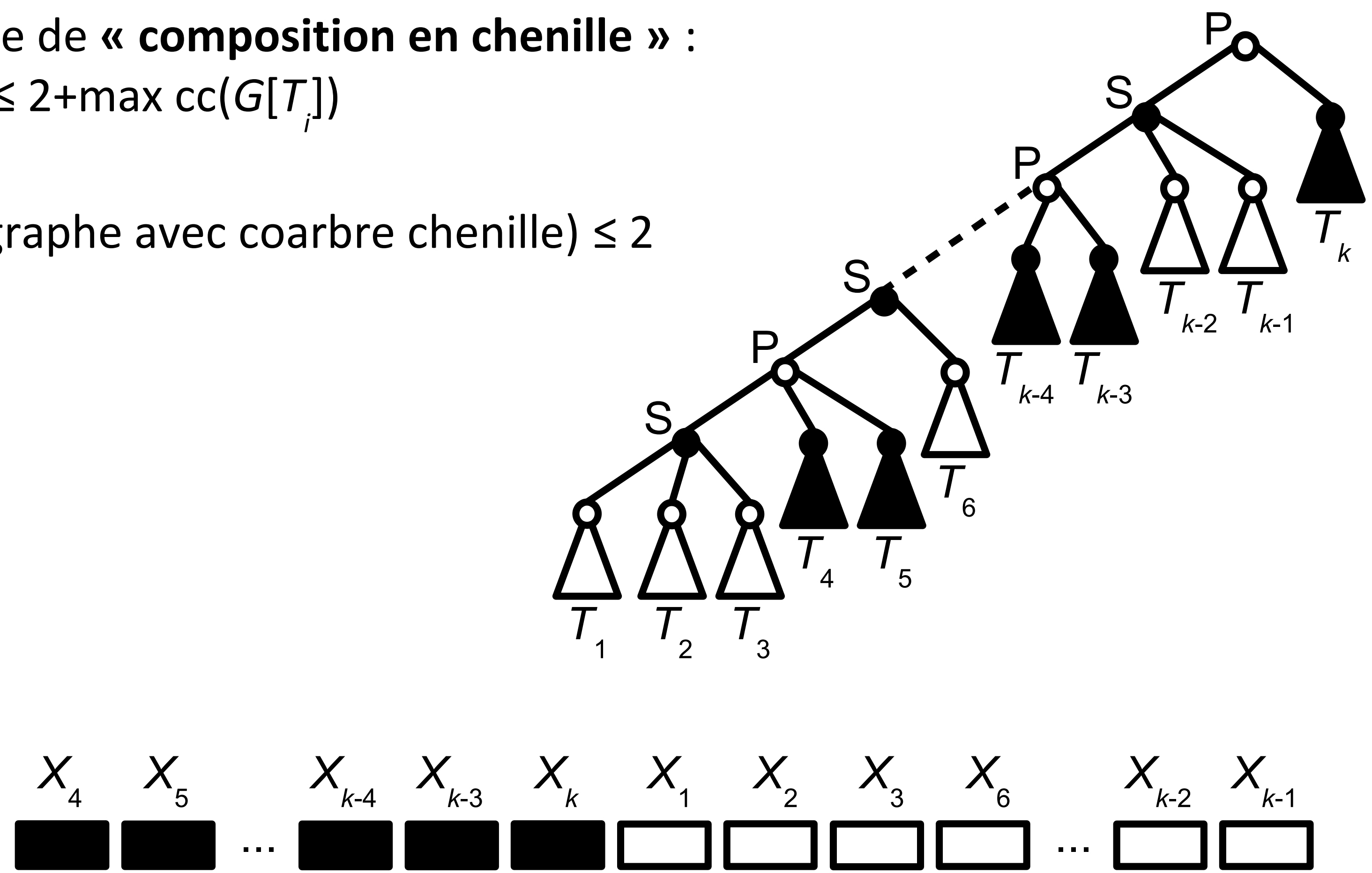
Contiguïté et linéarité des cographes

Bornes sur les cographes : linéarité et contiguïté en O(log n)

Crespelle & Gambette, 2012

Lemme de **« composition en chenille »** :
$cc(G) \leq 2 + \max cc(G[T_i])$

cc(cographe avec coarbre chenille) ≤ 2



Contiguïté et linéarité des cographes

Bornes sur les cographes : linéarité et contiguïté en O(log n)

Crespelle & Gambette, 2012

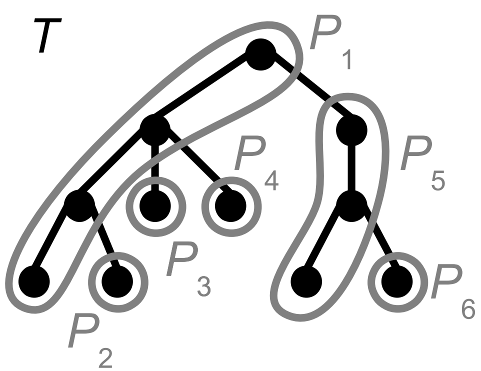
Lemme de **« composition en chenille »** :

$cc(G) \leq 2 + \max cc(G[T_i])$

Lemme de **partition d'un arbre en chemins** :

→ appliquer la « composition en chenille » sur les chemins

→ $cc(G) \leq 2 \log_2 n + 1$



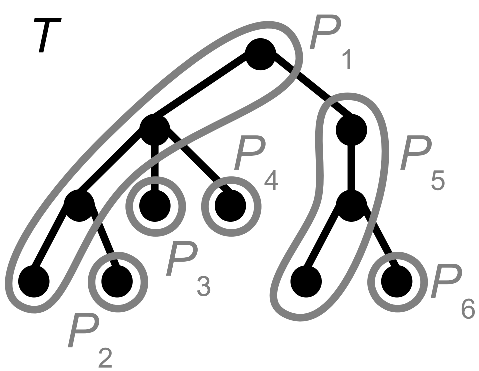
Contiguïté et linéarité des cographes

Bornes sur les cographes : linéarité et contiguïté en O(log n)

Crespelle & Gambette, 2012

Lemme de **« composition en chenille »** :
$cc(G) \leq 2+\max cc(G[T_i])$

Lemme de **partition d'un arbre en chemins** :
→ appliquer la « composition en chenille » sur les chemins
→ $cc(G) \leq 2 \log_2 n + 1$



Borne inférieure en $O(\log n)$ atteinte pour les cographes avec coarbre complet binaire

Perspectives

D'autres façons d'exploiter les **voisinages** algorithmiquement ?

- structure particulière permettant un codage efficace
- utilisation algorithmique du codage efficace

Etude approfondie des paramètres **contiguïté** et **linéarité** :

- Calcul sur certaines classes de graphes (cographes) ?
- Algorithmes d'optimisation quand ils sont bornés ?

Des questions ?

Merci pour votre attention !

Plus de détails dans :

- Christophe Crespelle et Philippe Gambette. Efficient Neighbourhood Encoding for Interval Graphs and Permutation Graphs and $O(n)$ Breadth-First Search, *Proceedings of the 20th International Workshop on Combinatorial Algorithms* (IWOCA'09), LNCS 5874, p. 146-157.

- Christophe Crespelle et Philippe Gambette. Unrestricted and Complete Breadth-First Search of Trapezoid Graphs in $O(n)$ Time, *Information Processing Letters* 110, p. 497-502.

- Christophe Crespelle et Philippe Gambette. A tight bound on the contiguity of cographs, en préparation, 2012.