Cours d'algorithmique des graphes du MPRI

Michel Habib
Rédigé par Philippe Gambette

23 janvier 2009

Table des matières

Chapitre 1

Introduction

Ces notes ont été prises à l'occasion des cours d'algorithmique des graphes du Master Parisien de Recherche en Informatique[1] en 2006/2007, donnés par Michel Habib (8 séances de 90 minutes).

1.1 Objectifs

Dans de nombreux domaines tels que la chimie, la biologie, les réseaux de télécommunications ou encore les réseaux sociaux des modèles à base de graphes sont utilisés quotidiennement en recherche. De même les graphes constituent des outils importants et très utilisés de modélisation en informatique. Pour s'en convaincre il suffit de considérer la place faite aux algorithmes sur les graphes dans les ouvrages classiques d'algorithmique (cf. par exemple l'excellent *Algorithm Design* [KT06]).

En effet malgré la simplicité apparente de leur définition, les graphes capturent une large part de la complexité algorithmique (i.e. des classes de complexité algorithmique telles P, NP et bien d'autres peuvent se définir comme l'ensemble des problèmes de graphes exprimables dans un certain fragment logique). Il est donc très important de bien comprendre la structure des graphes afin d'utiliser des modélisations pertinentes à base de graphes (i.e. des modélisations sur lesquelles les algorithmes de résolution sont efficaces).

Ce module a deux objectifs principaux : le premier concerne la compréhension de la complexité structurelle des graphes via des décompositions de graphes et les invariants de graphes associés, tandis que le deuxième est centré sur la conception d'algorithmes "efficaces" sur les graphes et l'étude des outils algorithmiques nécessaires.

1.2 Déroulement du cours en 2006/2007

- Semaine 1 : introduction, graphes et logique, applications de la théorie des graphes, classes de graphes, algorithmes robustes.
- Semaine 3 : modules (section 3.1), décomposition modulaire (section 3.2), familles partitives (section 3.3), graphes premiers (section 3.4), cographes (section 3.5).
- Semaine 4 : algorithmes de décomposition modulaire, historique (section 4), approche du module minimal (section 4.1), approche des graphes sandwichs (section 4.3), permutation factorisante (section 4.6).
- Semaine 5 : structures de données sur les graphes (section 2.3), décomposition modulaire et affinage de partition (section 4.4), algorithme LexBFS (section 5.1).

[1] http://mpri.master.univ-paris7.fr/C-2-29-1.html

- Semaine 6 : correction de l'exo 1, propriétés de LexBFS (section 5.2), calcul du diamètre (section 5.3), reconnaissance des graphes triangulés et d'intervalles (section 5.4).
- Semaine 7 : correction de l'exo 6 sur les graphes d'intervalles, arbre de cliques (section 6.1), k-arbres (section 6.2), décompositions arborescentes (section 6.3), mineurs (section 6.4).
- Semaine 9 : ordres d'intervalles (section 7.1), extensions linéaires (section 7.2), dimension d'un ordre (section 7.3), graphe de permutation (section 7.4), graphe trapézoïdal (section 7.5).
- Semaine 10 : examen.
- Semaine 11 : corrigé du partiel (Γ-classes, décomposition modulaire, et graphes de comparabilité, section 7.6).

1.3 Remerciements

Merci :
– à Marc Mezzarobba, Jean Daligault, Guyslain Naves, Géraldine Del Mondo, et Sylvain Raybaud pour leurs relectures,
– à Vincent Limouzy et Fabien de Montgolfier pour leurs fichiers BibTeX (le BibTex de Hans Bodlaender[2] a aussi été abondamment utilisé),
– à Matthieu Muffato pour les astuces sur la création des figures.

[2] http://www.cs.uu.nl/~hansb/graphs-bib/index.html

Chapitre 2

Les bases de théorie des graphes

2.1 Vocabulaire de base

Un **graphe orienté** ou **digraphe** $G = (V, E)$ est défini comme un ensemble V de n **sommets** ou **nœuds** (n est appelé l'**ordre** du graphe), et un ensemble $E \subset V^2$ de m **arcs** reliant ces sommets. Si deux sommets peuvent être reliés par plusieurs arcs, on parle de **multigraphe**. Un arc reliant le sommet v au sommet v' est noté (v, v').

Un **graphe** ou **graphe simple non orienté sans boucle** est un digraphe sans boucle[1] $G = (V, E)$ tel que pour tout arc (v, v') de E, l'arc orienté dans l'autre sens (v', v) appartient aussi à E. On parle alors d'**arêtes**, que l'on continue à noter (v, v'), en considérant que $(v, v') = (v', v)$.

Deux sommets reliés par une arête sont dits **adjacents** ou **voisins**. Le **voisinage** ou voisinage ouvert d'un sommet v, soit l'ensemble de ses sommets adjacents, est noté $\mathcal{N}(v)$. Sa taille est appelée **degré** du sommet v et notée $deg(v)$ et un sommet de degré nul est dit **isolé**. Un graphe n'ayant que des sommets de degré d est appelé graphe d-**régulier**.

Le **voisinage clos** d'un sommet v est défini comme $\mathcal{N}[v] = \mathcal{N}(v) \cup \{v\}$. Deux sommets v_1 et v_2 sont dits **vrais jumeaux** s'ils ont le même voisinage et sont adjacents, c'est-à-dire qu'ils ont le même voisinage clos $\mathcal{N}[v_1] = \mathcal{N}[v_2]$. Ainsi deux sommets v_1 et v_2 sont **faux jumeaux** s'ils ont le même voisinage mais ne sont pas adjacents, soit $\mathcal{N}(v_1) = \mathcal{N}(v_2)$. On définit aussi le non-voisinage de v comme étant $\overline{\mathcal{N}}(v) = V \backslash \mathcal{N}[v]$.

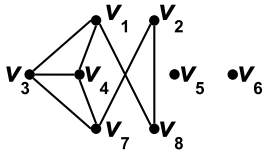


FIG. 2.1 – Le voisinage de v_1 est $\mathcal{N}(v_1) = \{v_3, v_4, v_8\}$, son degré est donc 3. Les sommets v_5 et v_6 sont isolés et les deux sont des faux jumeaux. v_3 et v_4 sont des vrais jumeaux. Le graphe a trois composantes connexes : $\{v_1, v_2, v_3, v_4, v_7, v_8\}$, $\{v_5\}$ et $\{v_6\}$.

Un graphe est **connexe** si pour tous sommets x et y, on peut atteindre y par une **marche** à partir de x, c'est à dire une suite de nœuds $(v_0 = x, v_1, \ldots, v_{k+1} = y)$ telle que v_i soit voisin de v_{i+1} pour tout i de $[\![0, k]\!]$. La connexité est une relation d'équivalence, dont les classes d'équivalences sont appelées **composantes connexes** du graphe G (une composante connexe est donc un ensemble maximal de sommets tel qu'on peut effectuer une marche d'un sommet à l'autre pour toute paire de sommets de cet ensemble, sommets qu'on dit alors **connectés**).

[1] Une **boucle** est un arc reliant un sommet v à lui-même : (v, v).

Le **complémentaire** d'un graphe non orienté G, noté $\overline{G}$, est le graphe de sommets V et d'arêtes $\{(v_1, v_2) \in V^2 / v_1 \neq v_2\} \backslash E$ (si deux sommets distincts v et v' sont adjacents dans G, ils ne le seront pas dans $\overline{G}$ et inversement). Remarquons que v_1 et v_2 sont faux jumeaux dans G si et seulement si ils sont vrais jumeaux dans $\overline{G}$.

Le sous-graphe de G induit par un sous-ensemble de sommets $S \subseteq V$, noté $G[S]$ est le graphe de sommets S et d'arêtes $E \cap S^2$, c'est-à-dire le graphe obtenu en gardant de G ses sommets dans S et toutes les arêtes les reliant. Soit G' un graphe. On dit qu'un graphe G est **sans** G' si aucun des sous-graphes de G n'est **isomorphe** (identique à réétiquetage des nœuds près) à G'. Un **graphe partiel** de $G = (V, E)$ est un graphe $H = (V, E')$ tel que $E' \subseteq E$ (on peut l'obtenir à partir de G par des effacements d'arêtes).

On note K_n la **clique**, c'est-à-dire le graphe **complet**, à n sommets et $\frac{n(n-1)}{2}$ arêtes (ou n-**clique**) : $\forall v \neq v' \in V, (v, v') \in E$. Son complémentaire (c'est-à-dire le graphe sans arêtes à n éléments) est noté S_n et appelé le **stable** à n éléments. On note P_n le **chemin**, ou la **chaîne** à n sommets, c'est-à-dire le graphe à n sommets $v_1, \ldots, v_n$ et $n-1$ arêtes (v_i, v_{i+1}), pour $1 \leqslant i < n$. Si l'on ajoute une n-ième arête (v_n, v_1), on obtient le **cycle** à n sommets C_n. En considérant les arêtes ci-dessus comme des arcs, on obtient la définition d'un **chemin orienté** et d'un **circuit**.

Un graphe (V, E) dont l'ensemble de sommets V peut être partitionné en $V_1 \cup V_2$ de telle sorte que V_1 et V_2 sont stables est un graphe **biparti**. Si de plus $E = V_1 \times V_2$, c'est-à-dire que toutes les arêtes entre V_1 et V_2 sont dans G, alors le graphe est **biparti complet**, et noté $K_{|V_1|,|V_2|}$.

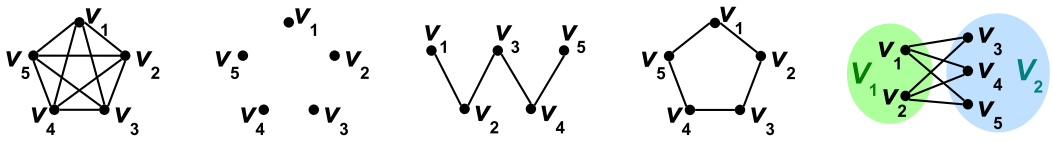


FIG. 2.2 – La clique K_5 (a), le stable S_5 (b), le chemin P_5 (c), le cycle C_5 (d), le graphe biparti complet $K_{2,3}$ (e).

Une **clique maximale** d'un graphe G est un sous-graphe induit $G[V']$, $V' \subseteq V$, tel que $G[V']$ est une clique et $\forall V'' \supset V'$, $G[V'']$ n'est pas une clique. Par abus de notation on écrira aussi parfois que l'ensemble de sommets lui-même, V', est une clique maximale de G. Attention à ne pas confondre la clique maximale avec la **clique maximum** qui désigne usuellement une clique de taille maximale d'un graphe.

2.2 Un petit exercice pour s'entraîner à manipuler tout ça

Exercice 1. Écrire un algorithme qui vérifie l'existence d'une clique à $\delta + 1$ sommets dans un graphe régulier de degré δ. L'étendre au cas où le graphe n'est pas δ-régulier, mais seulement de degré borné par δ.

On remarque que dans un graphe δ-régulier, une clique à $\delta+1$ sommets est une composante connexe à $\delta + 1$ sommets.

Si le graphe n'est pas régulier il suffit de commencer par filtrer l'ensemble de départ : on initialise S à l'ensemble des sommets du graphe de degré δ. Puis, tant qu'il existe $a \in S$ ayant un voisin hors de S, éliminer a de S.

Cet algorithme est en $O(n + m)$.

2.3 Graphes et structures de données

Il existe plusieurs façons de représenter un graphe en mémoire. Dans tous les articles d'algorithmique de graphes on fait l'hypothèse qu'on peut mettre dans un mot mémoire (qu'on compte en $O(1)$) le numéro d'un sommet (qui est donc en fait en $\log(|V|)$).

Voici quelques représentations possibles d'un graphe :

- la **liste d'adjacence** : $O(|V| + |E|)$ mots en mémoire. Le test d'adjacence se fait en $O(|\mathcal{N}(x)|)$ (on peut travailler avec des listes triées et donc s'arrêter quand on dépasse le numéro du sommet recherché).
- la **matrice d'adjacence** : $O(|V|^2)$ mots en mémoire. Le test d'adjacence est en $O(1)$.
- d'autres représentations personnalisées : un pointeur pour chaque arc...

Parfois la matrice d'adjacence ne peut être obtenue mais les listes d'adjacences pour chaque sommet, locales, peuvent être connues (typiquement dans le cas du graphe du web).

Exercice 2. Comment allier les avantages des deux représentations ? On veut accéder à un arc en $O(1)$, construire la représentation du graphe en $O(n + m)$, en s'autorisant un espace en $O(n^2)$.

Il est parfois non nécessaire de visiter tous les sommets (ou toutes les arêtes) une seule fois pour vérifier une propriété de graphe (par exemple pour certaines propriétés du complémentaire d'un graphe). Il existe donc très peu de bornes inférieures pour des problèmes sur les graphes.

Il est possible aussi de taguer les matrices d'adjacence des graphes orientés avec $2n$ bits supplémentaires, un pour chaque ligne et un pour chaque colonne, qui indiquent si la ligne ou la colonne a été complémentée. L'idée est de réduire ainsi significativement le nombre de 1 dans les matrices d'adjacence (ou la taille des listes d'adjacence), et donc obtenir des algorithmes en $O(n+m')$, avec $m' << m$, sur des graphes orientés [DGM02]. Ces représentations partiellement complémentées de graphes orientés sont illustrées en figure 2.3.

	1	2	3	4
1	1	1	1	0
2	0	0	1	0
3	1	0	1	1
4	1	0	0	0

→

	$\overline{1}$	2	$\overline{3}$	4
1	0	1	0	0
2	1	0	0	0
3	0	0	0	1
4	0	0	1	0

FIG. 2.3 – Représentation partiellement complémentée d'un graphe orienté : complémenter les colonnes 1 et 3 permet de baisser la taille des listes d'adjacence.

Chapitre 3

Introduction à la décomposition modulaire

3.1 Modules et partitions modulaires

Les graphes considérés dans la suite de ce chapitre sont finis, non orientés et sans boucle.

Définition 1. *Pour un graphe $G = (V, E)$, un **module** est un sous-ensemble $M \subseteq V$ de sommets ayant le même "voisinage extérieur" : $\forall x, y \in M, \mathcal{N}(x) \backslash M = \mathcal{N}(y) \backslash M$.*

Il existe une autre définition équivalente.

Définition 2. *Un ensemble M de sommets d'un graphe $G = (V, E)$ est un module ssi les sommets de M sont **indistingables** par les sommets extérieurs : $\forall v \in V \backslash M$,*
- *soit $M \subseteq \mathcal{N}(v)$ (tous les sommets de M sont adjacents à v),*
- *soit $M \cap \mathcal{N}(v) = \emptyset$ (aucun sommet de M n'est adjacent à v).*

Exemples de modules (illustrés en figure 3.1) :
- $\emptyset$, $\{v\}$ pour tout sommet v, et V, qui sont appelés modules **triviaux**,
- les composantes connexes d'un graphe ou de son complémentaire,
- tout sous-ensemble du graphe complet K_n ou du stable S_n.

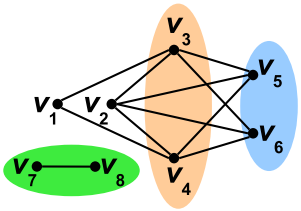


FIG. 3.1 – Exemple de modules : le sommet $\{v_1\}$, les deux composantes connexes $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ et $\{v_7, v_8\}$, ou encore les ensembles $\{v_3, v_4\}$, $\{v_5, v_6\}$, $\{v_1, v_2, v_5, v_6\}$,.

Un graphe est **premier** s'il n'a que des modules triviaux. Aucun graphe à trois sommets n'est premier. Le plus petit graphe premier est le chemin P_4 (son graphe complémentaire étant aussi P_4).

L'intérêt des modules est de pouvoir contracter tous les sommets qui ont le même comportement par rapport à l'extérieur.

Propriété 1. *M est un module de $G \Leftrightarrow M$ est un module de $\overline{G}$.*

Définition 3. *Le **casseur** d'un ensemble $A \subseteq V$ est un sommet $z \notin A$ tel que $\exists x, y \in A$ avec z et x adjacents, mais pas z et y, comme montré en figure 3.2.*

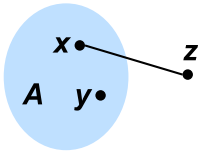


FIG. 3.2 – z est un casseur pour l'ensemble A contenant x et y. Pour mettre en relief le fait que z et y ne sont pas adjacents, on peut les relier par un trait discontinu.

Cette définition est motivée entre autres par le lemme suivant :

Lemme 1. *Si z est un casseur pour A alors tout module contenant A doit contenir z.*

Définition 4. *Un **module fort** M est un module qui ne chevauche pas un autre module : $\forall M'$ module, $M \backslash M' = \emptyset$ ou $M' \backslash M = \emptyset$ ou $M \cap M' = \emptyset$.*

Exercice 3. Écrire un algorithme linéaire qui calcule le graphe réduit correspondant aux deux opérations suivantes : contracter deux jumeaux, éliminer un sommet **pendant** (de degré 1).

Définition 5. *Une partition $\mathcal{P}$ de l'ensemble de sommets V d'un graphe G est une **partition modulaire** si toute partie est un module de V. On introduit donc le **graphe quotient** $G_{/\mathcal{P}}$ qui correspond au graphe G dont les ensembles de sommets de $\mathcal{P}$ ont été contractés en un sommet, illustré en figure 3.3(b).*

En particulier l'ensemble des singletons sommets est une partition modulaire d'un graphe.

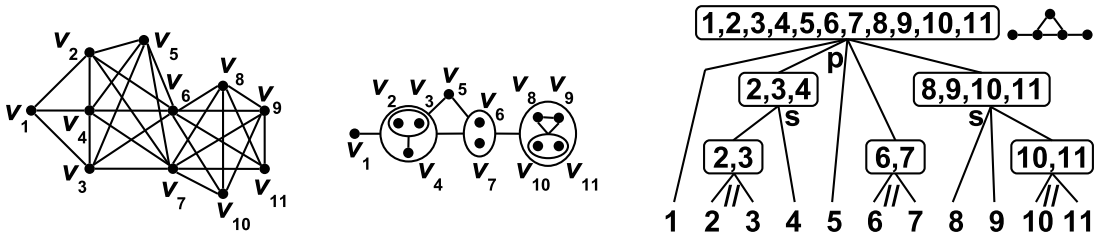
Propriété 2 ([MR84])**.** *Soit $\mathcal{P}$ une partition modulaire de G. L'ensemble de parties $\mathcal{X} \subseteq \mathcal{P}$ est un module de $G_{/\mathcal{P}}$ ssi $\bigcup_{M \in \mathcal{X}} M$ est un module de G.*

3.2 Décomposition modulaire

Théorème 1 ([Gal67, MP01, CHM81])**.** *Soit G un graphe ayant plus de 4 sommets, alors :*
- *soit le graphe est non connexe (c'est le cas **parallèle**).*
- *soit le graphe complémentaire est non connexe (c'est le cas **série**).*
- *soit $G_{/\mathcal{M}(\mathcal{G})}$ est un graphe premier, où $\mathcal{M}(G)$ est la partition modulaire contenant les modules forts maximaux (non triviaux) de G.*

On peut construire un algorithme de force brute en $O(n^3)$ pour calculer cette décomposition. De plus, on peut la représenter par un **arbre de décomposition modulaire** tel que :
- la racine correspond à V,
- les feuilles correspondent aux sommets,
- chaque nœud correspond à un module fort,
- il y a trois types de nœuds : parallèle, série, et premier,
- un noeud correspondant au module N est fils de celui correspondant au module M ssi M est le plus petit module fort tel que $N \subsetneq M$.

La décomposition modulaire a de nombreuses application :
- structure pour les graphes de comparabilité,
- encodage compact en utilisant la contraction de modules et si on garde dans tout nœud premier la structure du graphe premier,
- application possible d'un paradigme **diviser-pour-régner** pour résoudre des problèmes d'optimisation (par exemple pour le problème du stable maximum : résolution du stable maximum sur les nœuds premiers de l'arbre de décomposition, puis résolution du stable maximum pondéré sur le graphe quotient),



(a) (b) (c)

FIG. 3.3 – Un graphe (a), sa visualisation sous forme de graphe quotient (b), et son arbre de décomposition modulaire (c) : les nœuds séries sont étiquetés `s`, les nœuds parallèles `//` et le nœud premiers `p`.

– meilleure compréhension des algorithmes de graphes et de leurs structures de données.

3.3 Familles partitives

Propriété 3. *Si deux modules M et M' se chevauchent alors les ensembles suivants sont des modules :*

(i) $M \cup M'$,

(ii) $M \cap M'$,

(iii) $M \backslash M'$,

(iv) $M \Delta M'$[1].

Démonstration. La démonstration est facile, par disjonction de cas entre sommets intérieurs (appartenant à $M \cup M'$) ou extérieurs. □

Ces propriétés nous incitent à chercher un cadre plus général en définissant des objets qui les vérifient et généralisent donc la notion de décomposition modulaire.

Définition 6. *Soit $\mathcal{S} \subseteq \mathcal{P}(S)$, une famille de parties de S. $\mathcal{S}$ est une* ***famille partitive*** *si :*

– $S \in \mathcal{S}$, $\emptyset \in \mathcal{S}$ *et* $\forall x \in S, \{x\} \in \mathcal{S}$,

– *pour toute paire de parties $A, B \in \mathcal{S}$ qui se chevauchent :*

(i) $A \cup B \in \mathcal{S}$,

(ii) $A \cap B \in \mathcal{S}$,

(iii) $A \backslash B \in \mathcal{S}$ *et* $B \backslash A \in \mathcal{S}$,

(iv) $A \Delta B \in \mathcal{S}$.

Si la condition (iv) n'est pas vérifiée, on parle de ***famille faiblement partitive.***

La partition modulaire d'un graphe est donc une famille partitive par le lemme précédent (la partition modulaire d'un graphe orienté est une famille faiblement partitive) [CHM81].

De même que pour les modules, on dit que $X \in S$ est un élément fort de la famille $\mathcal{S}$ s'il ne chevauche aucun autre élément de $\mathcal{S}$. L'ensemble des éléments forts d'une famille partitive ou faiblement partitive peut alors être organisé en un arbre, où l'on peut distinguer certains noeuds **dégénérés** (ou **fragiles**), c'est-à-dire que pour tout sous-ensemble des fils de ce noeud, l'union des ensembles correspondant à ces noeuds appartient aussi à $\mathcal{S}$.

Théorème 2 ([CHM81])**.** *Les familles partitives admettent un arbre de décomposition avec deux types de nœuds : dégénérés ou premiers.*

[1] Δ étant la différence symétrique $M \Delta M' = M \backslash M' \cup M' \backslash M$

Théorème 3 ([CHM81]). *Les familles faiblement partitives admettent un arbre de décomposition avec trois types de nœuds : dégénérés, premiers, ou linéaires.*

3.4 Propriétés des graphes premiers

Théorème 4 (Caractérisation des graphes premiers). *Soit G un graphe premier ($|G| \geqslant 4$), alors G contient un P_4.*

On a même un théorème plus fort :

Théorème 5 ([CI98], p. 64-65). *Soit G un graphe premier, alors il a au plus un sommet non contenu dans un P_4.*

Démonstration. Supposons qu'il existe $v \in V$ qui n'appartienne pas à un P_4, alors le sous-graphe de G induit sur son non-voisinage, $G[\overline{\mathcal{N}}(v)]$, est un stable.

En effet, supposons par l'absurde que ce ne soit pas le cas, alors $G[\overline{\mathcal{N}}(v)]$ contient une composante connexe C. Comme G est premier alors C n'est pas un module de G, donc il existe un casseur α de C. On peut donc trouver deux sommets adjacents dans la composante connexe, v_1 et v_2 tels que par exemple (l'autre cas se traitant par symétrie), v_1 adjacent à α et que v_2 non-adjacent à α, comme illustré en figure 3.4. Remarquons alors que α est adjacent à v (si

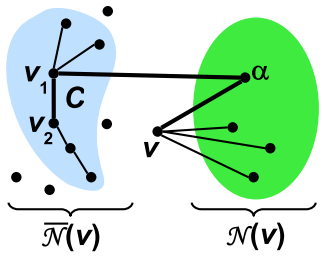


Fig. 3.4 – Illustration de la preuve qu'un sommet v n'appartenant pas à un P_4 ne peut avoir un non-voisinage qui ne soit pas un stable (sinon il est contenu dans le P_4 $v - \alpha - v_1 - v_2$).

ce n'était pas le cas, il appartiendrait à $\overline{\mathcal{N}}(v)$ et étant adjacent à v_1 il appartiendrait à C, et ne pourrait donc pas être casseur de C). Ainsi, le sous-graphe induit de G sur l'ensemble de sommets $\{v, \alpha, v_1, v_2\}$ est le P_4 $v - \alpha - v_1 - v_2$: contradiction !

Supposons maintenant qu'il existe deux sommets distincts v et v' n'appartenant pas à un P_4. Sans perte de généralité on peut considérer que v et v' sont non-adjacents (sinon on considère le graphe complémentaire), et donc $v' \in \overline{\mathcal{N}}(v)$. Comme $\{v, v'\}$ n'est pas un module de G, alors il existe un casseur β de cet ensemble. On considère que β est adjacent à v' et pas à v (l'autre cas se traitant par symétrie). v' et β sont donc tous deux non-adjacents à v, or on a montré que $G[\overline{\mathcal{N}}(v)]$ est un stable, et donc v' est n'est pas adjacent à β : contradiction ! □

En fait Cournier et Ille démontrent même que s'il existe un tel sommet v, alors v est adjacent aux deux sommets intérieurs d'un P_4 (G contient un sous-graphe à 5 sommets appelé **taureau**, illustré en figure 3.5, qui est, notons-le au passage, isomorphe à son complémentaire).

Fig. 3.5 – Un taureau

Théorème 6 (Schmerl, Trotter, Ille). *Soit G un graphe premier ($|G| = n > 4$), alors G contient un graphe premier à $n-1$ sommets ou un graphe premier à $n-2$ sommets.*

Pour $n \geqslant 7$ on a une version plus contrainte :

Théorème 7 ([ST93a]). *Soit G un graphe premier ($|G| = n \geqslant 7$), alors G contient un graphe premier à $n-2$ sommets.*

3.5 Les cographes

Définition 7. *La classe des **cographes** est la plus petite classe de graphes contenant le graphe à un point, et fermée par compositions série et parallèle*[2].

Le terme "cographe" vient de **completely reducible graph**, graphe complètement réductible, à l'aide des opérations séries et parallèles.

Théorème 8. *Un graphe est un cographe ssi il ne contient pas de P_4.*

Démonstration. Cette caractérisation se déduit du théorème sur les graphes premiers vu en section précédente. □

On peut en calculer en temps linéaire l'arbre de décomposition modulaire d'un cographe, qu'on appelle **coarbre** (voir chapitre 4.6). Les nœuds parallèles sont alors étiquetés par 0 et les nœuds séries par 1.

L'arbre de décomposition modulaire du complémentaire d'un graphe G est obtenu en inversant les 0 et 1 de l'arbre de décomposition modulaire de G. Pour savoir si deux nœuds du graphe sont adjacents, il suffit de vérifier que leur plus petit ancêtre commun dans l'arbre de décomposition modulaire est un nœud série.

Propriété 4. *G est un cographe ssi il existe un **ordre d'élimination par sommets jumeaux** défini comme suit : $\sigma = x_1 \dots x_n$ tel que $\forall i \in [\![1, n]\!]$, $\exists j \in [\![i, n]\!]$ tel que x_i et x_j sont des vrais jumeaux ou des faux jumeaux.*

Le premier algorithme de reconnaissance en temps linéaire des cographes est dû à Corneil, Pearl, et Stewart [CPS85], il est incrémental.

[2] Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux graphes. La **composition parallèle** de G_1 et G_2 est $(V_1 \cup V_2, E_1 \cup E_2)$, leur **composition série** est $(V_1 \cup V_2, E_1 \cup E_2 \cup \{(v_1, v_2)/v_1 \in V_1, v_2 \in V_2\})$.

Chapitre 4

Algorithmes de décomposition modulaire

Un historique non exhaustif d'algorithmes de décomposition modulaire est présenté en figure 4.1[1].

Nous présentons dans ce chapitre quelques approches principales utilisées pour la décomposition modulaire.

On pourra aussi se reporter au mémoire d'habilitation de Christophe Paul [Pau06] pour découvrir plusieurs approches algorithmiques à la décomposition modulaire.

4.1 L'approche par le module minimal

Propriété 5 (Module minimal contenant un ensemble)**.** *Pour tout $A \subset V$ il existe un unique module minimal contenant A.*

Démonstration. On considère tous les modules contenant A, leur intersection contient A et est un module. □

On en déduit un algorithme de force brute en $O(n^4)$. Pour toute paire de sommets (x, y), on calcule le module minimal $M(x, y)$ contenant x et y de façon incrémentale. Initialisons une variable `TempModule` avec $\{x, y\}$. Tant qu'il existe un casseur de `TempModule`, l'ajouter à `TempModule`. S'il n'existe pas de casseur de `TempModule`, `TempModule` est bien un module, et c'est $M(x, y)$!

La complexité de cet algorithme est en $O(n^2(n + m))$[2] et on peut en déduire un test de primalité (en vérifiant que le module minimal pour toute paire de sommets est V).

4.2 L'approche par la sous-modularité

Définition 8. *Soit V un ensemble fini. Une fonction f définie sur tous les sous-ensembles de V est* ***sous-modulaire*** *si $\forall X, Y \subset V, f(X) + f(Y) \geqslant f(X \cup Y) + f(X \cap Y)$*

En appelant $s(A)$ le nombre de casseurs d'un ensemble A, non vide, de sommets, alors s est une fonction sous-modulaire. C'est l'idée de base de l'algorithme d'Uno et Yagiura pour la décomposition modulaire des graphes de permutation en temps linéaire [UY00].

[1] D'après http://www.liafa.jussieu.fr/~fm/HistDM.html

[2] Pour arriver à cette complexité il faut utiliser une bonne structure de données, pour garder en mémoire en permanence l'intersection N des voisinages de tous les sommets de `TempModule`. À l'arrivée d'un candidat-casseur v, on vérifie (en $O(|\mathcal{N}(v)|)$) si son voisinage est égal à N : si c'est en effet un casseur, il fait diminuer la taille de N, sinon N reste inchangé.

Auteurs, références	Complexité	Commentaires
James, Stanton, Cowan [JSC72]	$O(n^4)$	premier algorithme polynomial
Maurer [Mau77]	$O(n^4)$	graphes orientés
Blass [Bla78]	$O(n^3)$	
Habib, Maurer [HM79]	$O(n^3)$	
Habib [Hab81]	$O(n^3)$	graphes orientés
Cunningham [Cun82]	$O(n^3)$	graphes orientés
Steiner [Ste82]	$O(n^3)$	
Spinrad [Spi82]	$O(n^2)$	
Buer, Möhring [BM83]	$O(n^3)$	
Corneil, Perl, Stewart [CPS85]	$O(n+m)$	cographes
Mc Creary [McC87]	$O(n^3)$	
Muller, Spinrad [MS89]	$O(n^2)$	incrémental
Adhar, Peng [AP90]	$O(\log^2 n)$, $O(nm)$ processeurs	parallèle, cographes, CRCW-PRAM
Lin, Olariu [LO91]	$O(\log n)$, $O(nm)$ processeurs	parallèle, cographes, EREW-PRAM
Cournier, Habib [CH93]	$O(n+m\alpha(m,n))$	
McConnell, Spinrad [MS94, MS97, MS99]	$O(n+m)$	
Cournier, Habib [CH94]	$O(n+m)$	
Ehrenfeucht, Gabow, McConnell, Sullivan [EGMS94]	$O(n^2)$	2-structures
Ehrenfeucht, Harju, Rozenberg [EHR94]	$O(n^2)$	2-structures, incrémental
Habib, Huchard, Spinrad [HHS95]	$O(n+m)$	graphes d'héritage
McConnell [McC95]	$O(n^2)$	2-structures, incrémental
Bonizzoni, Della Vedova [BV95, BV99]	$O(n^{3k-5})$	hypergraphe de dimension k
Morvan, Viennot [MV96]		parallèle
Capelle, Habib [CH97]	$O(n+m)$	si permutation factorisante fournie
Dahlhaus, Gustedt, McConnell [DGM97, DGM01]	$O(n+m)$	
Habib, Paul, Viennot [HPV99]	$O(n+m\log n)$	calcule une permutation factorisante
Uno, Yagiura [UY00]	$O(n)$	graphes de perm. si réalisation connue
McConnell, Spinrad [MS00]	$O(n+m\log n)$	
Habib, Paul [HP01, HP05]	$O(n+m\log n)$	cographes
Capelle, Habib, Montgolfier [CHdM02]	$O(n+m)$	graphes orientés, si perm. fact. fournie
Dahlhaus, Gustedt, McConnell [DGM02]	$O(n+m)$	graphes orientés
Bretscher, Corneil, Habib, Paul [BCHP03, BCHP08]	$O(n+m)$	cographes, LexBFS
Habib, Montgolfier, Paul [HdMP04]	$O(n+m)$	calcule une permutation factorisante
Shamir, Sharan [SS04]	$O(n+m)$	cographes, dynamique
McConnell, Montgolfier [MdM05]	$O(n+m)$	graphes orientés
Bergeron, Chauve, Montgolfier, Raffinot [BCdMR05]	$O(n)$	graphes de perm. si réalisation connue
Tedder, Corneil, Habib, Paul [TCHP08]	$O(n+m)$	

FIG. 4.1 – Historique de la décomposition modulaire

4.3 L'approche par le graphe sandwich

Définition 9 ([GKS95]). *Soient deux graphes $G_1 = (V, E_1)$ et $G_2 = (V, E_2)$ non orientés tels que $E_1 \subseteq E_2$ et soit π une propriété. Un graphe* ***sandwich*** *de G_1 et G_2 est un graphe $G_s = (V, E_s)$, tel que $E_1 \subseteq E_s \subseteq E_2$ (les arêtes de G_1 sont obligatoires et celles de $\overline{G_2}$ sont interdites). Le problème du graphe sandwich consiste à déterminer s'il existe un graphe sandwich de G_1 et G_2 satisfaisant la propriété π.*

Toutefois de nombreux problèmes de graphes sandwichs sont NP-complets, en particulier quand la propriété π est :

– G_s est un graphe triangulé[3].
– G_s est un graphe de comparabilité[4].
On a quand même des cas polynomiaux, quand la propriété π est :
– G_s est un cograph [GKS95].
– G_s admet un module non trivial [CEdFK98, SMTW01, HLP03].
Remarquons qu'on peut généraliser la notion de casseur aux problèmes de graphes sandwichs :

Définition 10. *Pour un sous-ensemble $A \subset V$, un casseur est un sommet $z \notin A$ tel que $\exists x, y \in A$ avec $(z, x) \in E_1$ et $(z, y) \notin E_2$.*

Le calcul du module sandwich minimal peut alors être fait en $O(n^2(n + m_1 + m_3))$ (avec $m_1 = |E_1|$ et $m_3 = |\overline{E_2}|$). Il est difficile d'obtenir une meilleure complexité avec cette idée.

4.4 L'approche par l'affinage de partitions

La technique d'affinage de partition a été introduite dans les années 70 pour des problèmes d'isomorphismes de graphes [CG70] ou de minimisation d'automates [Hop71, CC82], ou le tri des chaînes de caractères [PT87]. L'algorithme de tri rapide Quicksort [Hoa61] peut aussi être considéré comme un schéma d'affinage de partition.

Définition 11. *Soient $P = \{X_1, \ldots, X_n\}$ et $Q = \{Y_1, \ldots, Y_m\}$ deux partitions de V.*
*On introduit une relation d'ordre $\preceq$ pour définir le **treillis des partitions** : $P \preceq Q$ ssi $\forall j \in [\![1, m]\!]$, $\exists I \subseteq \{1, n\}/\ Y_j = \bigcup_{i \in I} X_i$ (chaque partie de Q est une union de parties de P).*

A quoi correspondent inf et sup pour cette relation $\preceq$? On peut définir l'inf mathématiquement : $\inf(P, Q) = \{A \cap B / A \in P, B \in Q\}$.
Notons que $A \cap B$ peut être vide, l'ensemble vide n'est en fait pas ajouté à la partition $\inf(P, Q)$.
Exemples : soient $P = \{1, 5, 7|2, 3|6, 8|4\}$ et $Q = \{1, 5, 6|2, 3, 4|7|8\}$, alors $\inf(P, Q) = \{1, 5|2, 3|4|6|7|8\}$ et $\sup(P, Q) = \{1, 5, 6, 7, 8|2, 3, 4\}$.

Définition 12. *L'**affinage d'une partition** $P = \{X_1, \ldots, X_n\}$ d'un ensemble V par le **pivot** $S \subseteq V$ se définit de la façon suivante : $Q = Affine(S, P) = \{X_1 \cap S, X_1 \setminus S, \ldots, X_n \cap S, X_n \setminus S\}$.*

Remarquons à nouveau que l'ensemble vide n'est pas ajouté à Q, et que $Q \preceq P$.
Exemple : Affine$(\{5, 6\}, \{1, 5, 7|2, 3|6, 8|4\}) = \{1, 7|2, 3|4|5|6|8\}$.

Propriété 6. *$Affine(S, P) = P$ ssi S est une union de parties de P.*

Propriété 7 (Dualité). *$Affine(S, P) = Affine(\overline{S}, P)$.*

Un affinage peut être calculé en $O(|S|)$ avec une bonne structure de données (liste doublement chaînée, généralisation de la structure utilisée pour résoudre en temps linéaire le problème du drapeau hollandais [Dij76][5]). Plus précisément, cette structure consiste en un stockage des

[3] Un graphe est **triangulé** ssi tout cycle de longueur $\geqslant 4$ contient une corde (le graphe est sans C_k pour $k \geqslant 4$), voir section 5.4.1

[4] Un graphe est un **graphe de comparabilité** ssi ses arêtes peuvent être orientées transitivement : si le graphe orienté résultat contient l'arête $a \longrightarrow b$ et $b \longrightarrow c$ alors il contient aussi l'arête $a \longrightarrow c$, voir aussi section 7.

[5] On a en entrée un tableau avec n bandes de tissu de trois couleurs différentes (bleu, blanc, rouge) qu'il faut ranger par couleurs, afin de reconstituer le drapeau hollandais (dans l'ordre : rouge, blanc, bleu). Pour cela il suffit d'utiliser un algorithme glouton et trois pointeurs sur

éléments de V dans une liste doublement chaînée, tel que tous les éléments consécutifs apparaissent dans la même partie de la partition. Comme montré en figure 4.2, chaque élément a un pointeur vers la partie qui le contient, et chaque partie a un pointeur vers son premier élément, et son dernier élément, dans la liste.

Exemple : la figure 4.2 illustre en détails l'algorithme permettant d'effectuer un affinage de partition en $O(|S|)$, avec $V = x_1, \ldots, x_7$, $P = \{C_1, C_2, C_3\}$, et $S = \{x_3, x_4, x_5\}$, où $C_1 = \{x_1, x_3, x_5\}$, $C_2 = \{x_2, x_7\}$, et $C_3 = \{x_4, x_6\}$.

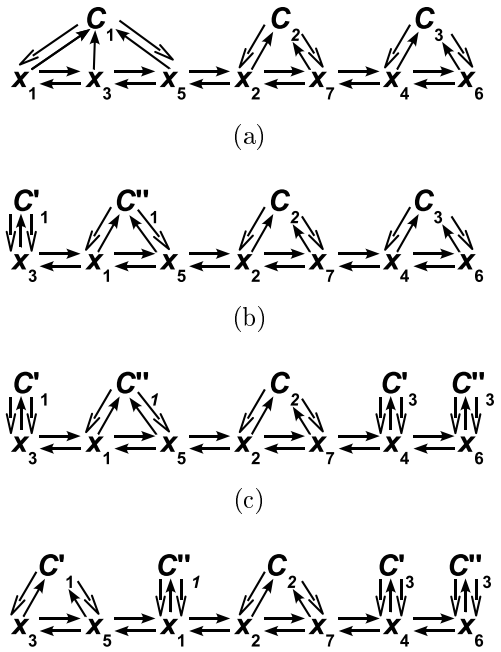


FIG. 4.2 – On traite successivement chaque élément de S : tout d'abord x_3 qui découpe la partie C_1 (b), puis x_4 qui découpe C_3 (c), et enfin x_5 qui réorganise C'_1 et C''_1 (d).

Exercice 4 (Ordre doublement lexicographique). Trouver un algorithme efficace pour ordonner lexicographiquement les lignes et les colonnes d'une matrice.

Exemple :

$$\begin{pmatrix} & C_1 & C_2 & C_3 & C_4 & C_5 \\ L_1 & 1 & 0 & 1 & 1 & 0 \\ L_2 & 0 & 1 & 0 & 0 & 1 \\ L_3 & 1 & 1 & 1 & 0 & 0 \\ L_4 & 1 & 1 & 0 & 0 & 0 \\ L_5 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \xrightarrow[\text{tri doublement lexicographique}]{} \begin{pmatrix} & C_3 & C_5 & C_2 & C_4 & C_1 \\ L_5 & 0 & 0 & 0 & 1 & 1 \\ L_4 & 0 & 0 & 1 & 0 & 1 \\ L_2 & 0 & 1 & 1 & 0 & 0 \\ L_1 & 1 & 0 & 0 & 1 & 1 \\ L_3 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Un tel ordre existe toujours, et pour les matrices d'incidence de graphes non dirigés, il donne un ordre sur les sommets aux propriétés intéressantes.

Le meilleur algorithme connu [PT87] pour une matrice à n lignes, m colonnes, et e valeurs non nulles, est en $O(L \log L)$ où $L = n + m + e$, et il utilise l'affinage de partition. L'argument

---

des cases du tableau : la case du bloc rouge la plus à droite, la première case non rangée (juste en dessous de la case du bloc blanc la plus à droite) et la case du bloc bleu la plus à gauche. A tout moment de l'algorithme, en notant ? une case non rangée, R, W et B une case du bloc respectivement rouge, blanc, ou bleu, le tableau est de la forme : R...RW ...W ?... ?B...B. Ainsi, on considère la case non encore rangée la plus à gauche et selon sa couleur, on la range dans son bloc (à droite du rouge, à droite du blanc, ou à gauche du bleu) en déplaçant la case qui était à sa place, et en mettant à jour les pointeurs.

d'optimalité de l'article est faux, on peut donc encore espérer trouver un algorithme linéaire.

Exercice 5 (Les jumeaux).
Écrire un algorithme linéaire simple qui calcule toutes les classes de sommets jumeaux d'un graphe G fourni en entrée.

Algorithme 1 (Recherche des jumeaux).
Partir avec $P = \{V\}$
Pour tout sommet $x \in V$ *faire :*
– $P = Affine(\mathcal{N}(x), P)$.

A la fin de l'algorithme 1, aucune partie de la partition ne peut être cassée par un sommet. Les parties de P sont constituées de faux jumeaux.

Pour les obtenir les vrais jumeaux, faire la même chose sur le graphe complémentaire.

4.5 L'approche d'Ehrenfeucht et al.

Le principe de l'algorithme d'Ehrenfeucht et al. [EHR94] consiste à :
– calculer la partition $\mathcal{M}(G, v)$ formée d'un sommet $\{v\}$ et de tous les modules maximaux ne contenant pas v,
– calculer la décomposition modulaire du graphe quotient $G_{/\mathcal{M}(G,v)}$,
– pour tout ensemble $X \in \mathcal{M}(G, v)$, calculer la décomposition modulaire de $G[X]$.

L'algorithme original est en $O(n^2)$ mais peut-être amélioré grâce à l'affinage de partition pour obtenir du $O(n+m\log n)$ [MS00], voire du $O(n+m\alpha(n,m))$, ou encore un temps linéaire avec une implémentation plus compliquée [DGM01].

4.6 L'approche par les permutations factorisantes

Définition 13 ([CH97]). *Une* ***permutation factorisante*** *d'un graphe* $G = (V, E)$ *est une permutation de* V *telle que tout module fort de* G *est un facteur de la permutation.*

L'idée est qu'on peut calculer une permutation factorisante d'un graphe en $O(n+m\log(n))$ [HPV99]. Elle permet alors de retrouver l'arbre de décomposition modulaire en temps linéaire [CHdM02].

Pour la première étape de calcul de la permutation, l'algorithme consiste à partir de la partition $\{\mathcal{N}(v), v, \overline{\mathcal{N}(v)}\}$, et à l'affiner récursivement, en utilisant comme pivot un sommet du graphe v et son voisinage (cet affinage se fait en $O(deg(v))$). Pour les cographes, cette étape peut se faire en temps linéaire [HP01, Pau06], ce qui donne un algorithme linéaire de reconnaissance des cographes. Mentionnons toutefois qu'il existe un autre algorithme linéaire simple de reconnaissance des cographes fonctionnant en deux parcours LexBFS[6] [BCHP03, BCHP08].

[6] LexBFS : voir chapitre suivant.

Chapitre 5

Parcours en largeur lexicographique

Un rapport de synthèse sur les applications de l'algorithme LexBFS a été présenté par Derek Corneil à la conférence WG 2004 [Cor04].

5.1 Algorithme LexBFS

Algorithme 2 (LexBFS, parcours en largeur lexicographique)**.**
Données : un graphe $G = (V, E)$ *et un sommet source* s
Résultat : un ordre total σ *de* V*.*
Affecter l'étiquette $\emptyset$ *à chaque sommet.*
`label`$(s) := \{n\}$
Pour i *de* n *à 1 faire :*
– *Choisir un sommet* v *d'étiquette lexicographique maximale.*
– $\sigma(i) := v$
– *Pour chaque sommet non numéroté* w *de* $\mathcal{N}(v)$ *faire :*
  – `label`$(w) :=$ `label`$(w) \cup \{i\}$ *// Concaténation de* i *à la fin de l'étiquette de* w

Cet algorithme, illustré sur un exemple en figure 5.1, peut être implémenté comme le propose Golumbic avec un tas [Gol80], mais il est plus simple de le considérer comme un affinage de partition :
$\mathcal{P}_0 = \{V\}$ (initialement tous les sommets ont une étiquette vide, et on va commencer le parcours depuis le sommet x, puis le sommet $y \dots$)
$\mathcal{P}_1 = \{\{x\}, \quad \mathcal{N}(x), \quad \overline{\mathcal{N}}(x)\}$
Etiquettes : $\{x\}$ $\emptyset$
$\mathcal{P}_2 = \{\{x\}, \{y\}, \quad \mathcal{N}(x) \cap \mathcal{N}(y), \quad \mathcal{N}(x) - \mathcal{N}(y), \quad \overline{\mathcal{N}}(x) \cap \overline{\mathcal{N}}(y), \quad \overline{\mathcal{N}}N(x) - \overline{\mathcal{N}}(y)\}$
Etiquettes : $\{x, y\}$ $\{x\}$ $\{y\}$ $\emptyset$
$\dots$

5.2 Propriétés du parcours

Propriété 8. *Les propriétés suivantes montrent l'intérêt du parcours LexBFS :*
– *si* G *est triangulé, le dernier sommet visité est simplicial*[1]*,*
– *si* G *est un cograph, le dernier sommet visité est un jumeau,*
– *si* G *est un graphe d'intervalles*[2]*, la dernière clique maximale visitée peut être choisie comme clique extrême dans une chaîne de cliques maximales,*

[1] Un sommet est **simplicial** si son voisinage est une clique.
[2] Un graphe d'intervalles est un graphe d'intersection d'intervalles de réels, voir section 5.4.2.

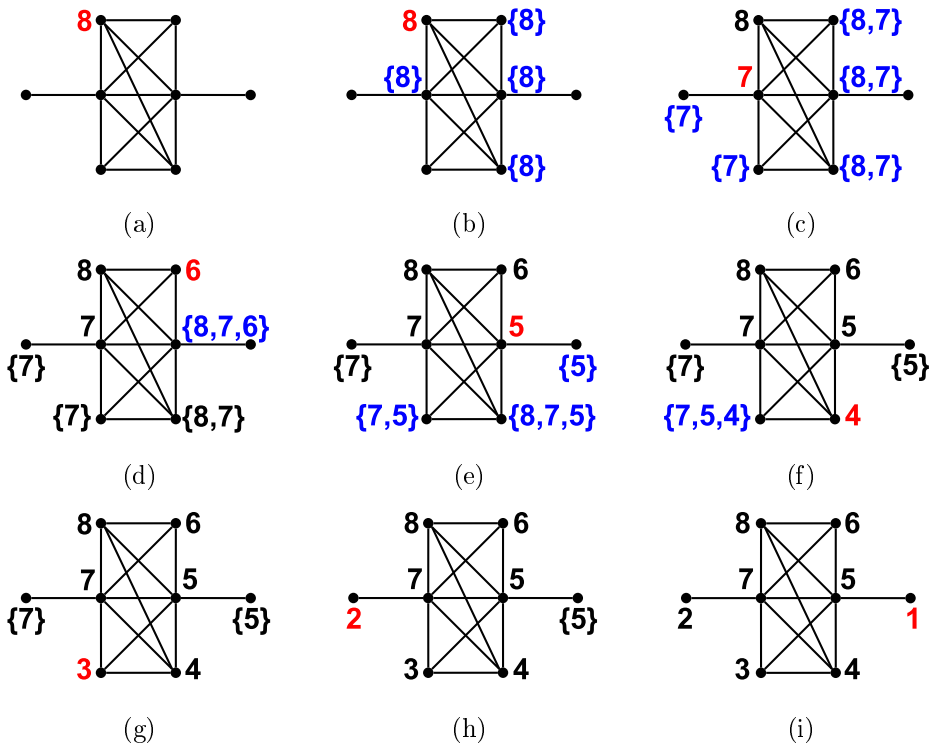


FIG. 5.1 – Déroulement de l'algorithme LexBFS sur un exemple

– *si G est un graphe de comparabilité, alors le dernier sommet visité par un LexBFS sur $\overline{G}$ peut être choisi comme source d'une orientation transitive de G.*

Les propriétés du parcours LexBFS sont particulièrement intéressantes sur les classes de graphes **héréditaires** (quand on enlève un sommet au graphe, il reste dans la classe). De plus, les preuves sont aussi instructives, car pas naturelles, puisque les propriétés ne sont pas des propriétés d'invariant.

5.3 Calcul du diamètre d'un graphe

Définition 14. *Soit G un graphe non orienté. On appelle **distance** entre v et v', qu'on note $dist(v, v')$, la longueur du plus court chemin entre v et v' et on définit les propriétés suivantes de G :*

– *l'**excentricité** d'un sommet c de G : $exc(c) = \max_{v \in V} dist(c, v)$,*

– *le **diamètre** du graphe : $diam(G) = \max_{v \in V} exc(v)$,*

– *le **rayon** du graphe : $rayon(G) = \min_{v \in V} exc(v)$.*

Le diamètre d'un réseau est lié à la qualité de service, il indique le plus mauvais cas possible. C'est un paramètre de nombreux algorithmes distribués (par diffusion). Il est pourtant difficile à calculer : il est plus facile d'en trouver une borne inférieure, sur les réseaux, en exhibant simplement deux sommets à distance égale à cette borne inférieure.

Théorème 9 ([Jor69]). *Un arbre admet un ou deux centres (suivant la parité du diamètre), sommets par lesquels passent toutes les chaînes élémentaires de longueur maximum.*

En outre pour un arbre T, $rayon(T) = \lceil \frac{diam(T)}{2} \rceil$.

Le diamètre se calcule en $O(n)$ sur les arbres, par un algorithme *bottom-up* (en procédant par étapes successives d'effacement des feuilles).

L'algorithme naïf pour les graphes consiste à calculer l'excentricité de chaque sommet par n parcours en largeur, ce qui a une complexité en $O(mn)$. Pour être plus efficace, on peut rechercher plutôt un algorithme qui calcule l'ensemble des plus courtes distances sur le graphe, par une méthode à base de multiplications de matrices booléennes, ou de parcours dans le graphe.

Intéressons-nous au problème de certificat du diamètre : faut-il fournir toute la matrice des $dist(v, v')$? Si l'on fournit seulement une arborescence des plus courtes chaînes issues d'un sommet v, on a seulement un encadrement : $exc(v) \leqslant diam(G) \leqslant 2exc(v)$ (l'inégalité de droite venant du fait qu'on peut passer par v pour aller d'un sommet du graphe à un autre). En fait, un certificat du fait que le diamètre de G est k est un graphe partiel $H \subset G$ de diamètre k (si possible tel que son diamètre se calcule linéairement).

L'algorithme LexBFS permet de donner de bonnes approximations du diamètre sur certaines classes de graphes.

Théorème 10 ([CDHP01]). *Deux parcours LexBFS successifs permettent de calculer le diamètre des arbres et de l'approcher à un près sur les graphes triangulés.*

L'algorithme consiste à effectuer un premier parcours LexBFS du graphe, qui se termine sur un sommet u utilisé comme source du second parcours LexBFS qui se termine sur un sommet v : on a alors $diam(G) = dist(u, v)$ ou $diam(G) = dist(u, v) + 1$.

Ce système de répétitions successives de parcours LexBFS constitue une excellente heuristique pour trouver une approximation du diamètre de grands graphes quelconques.

L'idée de cette approximation à 1 près permet d'établir un lien avec un autre problème : parmi une famille de parties $\{X_1, \ldots, X_m\}$ d'un ensemble à n éléments telle que $\sum_{1 \leqslant j \leqslant n} |X_j| = k$, trouver deux ensembles X_i et X_j disjoints peut-il se faire en $O(n + m + k)$? La relation avec le problème de diamètre est illustrée en figure 5.2.

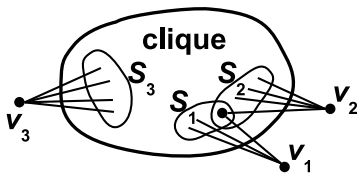


FIG. 5.2 – Le graphe ci-dessus est constitué d'une clique, ainsi que trois sommets v_1, v_2 et v_3 non adjacents deux à deux, et adjacents respectivement à des sommets de la clique : S_1, S_2, S_3. Si on peut trouver deux S_i disjoints, par exemple $S_1 \cap S_3 = \emptyset$, alors le diamètre est 3, le diamètre est 2 sinon.

Avec un simple parcours en largeur il est aussi possible d'approximer le diamètre :

Théorème 11 ([CDK03]). *Si G ne contient pas de cycle sans corde de longueur supérieure ou égale à k, alors le parcours en largeur permet de trouver un sommet v tel que $ecc(v) \geqslant diam(G) - \lfloor k/2 \rfloor$.*

Démonstration. On ne donne pas la preuve, mais quelques propriétés du parcours en largeur qui y sont utiles. Soit $\sigma \in \mathcal{S}_n$, l'ordre de visite des sommets d'un graphe G à n nœuds par un parcours en largeur de source s ($\sigma(s) = 1, \max \sigma(v) = n$). Le parcours en largeur pouvant être vu comme un arbre couvrant de G de racine s, pour tout sommet $v \neq s$ on appelle $p(v)$ le père du noeud v dans cet arbre. On a alors les propriétés suivantes :

– si $dist(s,y) = q > 0$, alors $dist(s,p(y)) = q-1$ et $\forall v \neq p(y) \in \mathcal{N}(y)/dist(s,v) = q-1$, $\sigma(v) > \sigma(p(y))$,
– si $dist(s,x) < dist(s,y)$ alors $\sigma(x) < \sigma(y)$,
– si $dist(s,x) = dist(s,y)$ et $\sigma(x) < \sigma(y)$, alors soit $\sigma(p(x)) < \sigma(p(y))$, soit $p(x) = p(y)$.
– si $dist(s,x) = dist(s,y) = dist(x,z)$, $\sigma(x) < \sigma(y) < \sigma(z)$ et $p(x)$ adjacent de z, alors $p(x) = p(y) = p(z)$ (et en particulier $p(x)$ adjacent de y).

□

Théorème 12 ([CDK03]). *Si v est le dernier sommet d'un parcours en largeur d'un graphe triangulé G, alors $ecc(v) \geqslant diam(G) - 1$.*

Théorème 13 ([CDK03, Dra05]). *Si v est le dernier sommet d'un parcours en largeur d'un graphe d'intervalles G, alors $ecc(v) \geqslant diam(G) - 1$. Si le parcours en largeur de G commence par une source s telle que $\mathcal{N}(s) \neq V$, alors parmi les noeuds du dernier niveau du parcours en largeur (ceux tels que la distance $dist(s,v)$ est maximale), il en existe un dont l'excentricité est égale au diamètre de G.*

Ainsi on obtient un théorème général liant parcours en largeur et diamètre sur plusieurs classes de graphes.

Théorème 14 ([CDK03]). *Deux parcours en largeur successifs suffisent à calculer le diamètre des arbres et l'approcher à 1 près sur les graphes triangulés, les graphes sans triplet astéroïde...*

5.4 Algorithmes de reconnaissance

5.4.1 Reconnaissance des graphes triangulés

Rappelons la définition d'un graphe triangulé, avant d'introduire celle d'un ensemble séparateur, en vue d'obtenir des caractérisations des graphes triangulés.

Définition 15. *Un graphe est **triangulé** ssi tout cycle de longueur $\geqslant 4$ contient une corde (le graphe est sans C_k pour $k \geqslant 4$).*

Définition 16. *Le sous-ensemble de sommets $S \subset V$ est un **séparateur** de G s'il existe a et b non adjacents dans G et non connectés dans $G[V \backslash S]$. On dit alors aussi que c'est un a,b-séparateur (voir figure 5.3). Si S minimal pour l'inclusion parmi tous les séparateurs de G, c'est un **séparateur minimal** de G.*

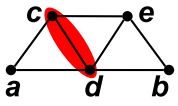


FIG. 5.3 – $\{c,d,e\}$ est un a,b-séparateur de G, $\{c,d\}$ ou $\{d,e\}$ sont des a,b-séparateurs minimaux de G.

Propriété 9 ([Dir61]). *Dans tout graphe triangulé G, tout séparateur minimal de G est une clique.*

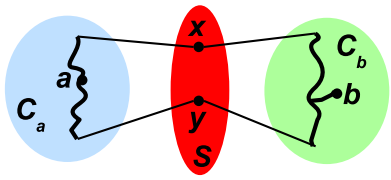


FIG. 5.4 – Tout séparateur minimal S d'un graphe triangulé est une clique : s'il n'y avait pas d'arête entre x et y, le graphe aurait un cycle de longueur supérieure ou égale à 4.

Démonstration. Soit S un séparateur minimal de G, alors il existe deux sommets a et b tels que S est un a,b-séparateur de G. On appelle C_a (respectivement C_b) la composante connexe de $G[V \backslash S]$ contenant a (respectivement b).

Soient deux sommets x et y dans S, supposons par l'absurde qu'ils ne sont pas adjacents.

Considérons alors le plus petit cycle de G contenant x, y, au moins un sommet de C_a et au moins un sommet de C_b. Comme S est minimal, x est adjacent d'un sommet de C_a et d'un sommet de C_b. De même, y est adjacent d'un sommet C_a et d'un sommet de C_b, comme illustré en figure 5.4. Et aucun sommet de C_a n'est adjacent d'un sommet de C_b donc un tel cycle existe et a une longueur supérieure ou égale à 4 (il doit passer successivement par x, C_a, y et C_b).

Or G est un graphe triangulé donc ce cycle contient une corde, l'arête (x, y) (par minimalité du cycle et puisqu'un sommet de C_a ne peut être adjacent d'un sommet de C_b) : contradiction ! □

Cette propriété est utile pour la démonstration des lemmes suivants, qui permettent d'aboutir au théorème 15 de caractérisation des graphes triangulés à l'aide du parcours LexBFS.

Lemme 2 ([Dir61]). *Tout graphe triangulé possède au moins un sommet simplicial.*

Démonstration. Soit un graphe triangulé $G = (V, E)$, on procède par induction sur $n = |V|$ pour prouver la propriété suivante qui est plus forte que le lemme : tout graphe triangulé possède au moins un sommet simplicial, et si ce n'est pas une clique alors il en possède au moins deux non adjacents.

L'initialisation étant triviale pour $n = 1$, supposons que tout graphe triangulé à moins de n sommets en possède un simplicial, et démontrons que G ayant n sommets en a un aussi.

Si G est une clique, alors tout sommet est simplicial. Si ce n'est pas le cas, alors il possède deux sommets non adjacents a et b. Soit S un a,b-séparateur minimal de G. $G[V \backslash S]$ a un certain nombre de composantes connexes (voir figure 5.5), appelons A (respectivement B) l'ensemble des sommets de celle qui contient a (respectivement b). Montrons que A contient un sommet simplicial (on procède de même pour B).

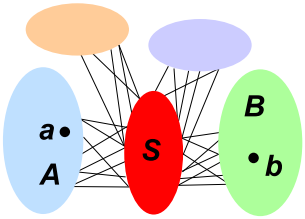


FIG. 5.5 – Les deux composantes connexes A et B contenant respectivement a et b séparées par un a,b-séparateur minimal S de G contiennent chacune un sommet simplicial.

– Soit $G[A \cup S]$ est une clique. Alors a est simplicial dans $G[A \cup S]$. Or tous les voisins de a appartiennent à $G[A \cup S]$, donc a est simplicial dans G.

– Soit $G[A \cup S]$ n'est pas une clique. Comme b n'appartient pas à $A \cup S$, $G[A \cup S]$ a moins de n sommets, donc on applique l'hypothèse de récurrence : $G[A \cup S]$ a deux sommets simpliciaux non adjacents qu'on appelle x et y. Trois cas se présentent alors :
  – soit $x \in A$, alors tous ses voisins sont dans $A \cup S$, donc x est aussi simplicial dans G.
  – soit $y \in A$, alors de même y est simplicial dans G.
  – soit $x \in S$ et $y \in S$, x et y ne sont pas adjacents, mais S est un séparateur minimal d'un graphe triangulé donc ce devrait être une clique d'après la propriété 9 : contradiction !

Finalement, dans tous les cas, on a montré que A et B contiennent chacun au moins un sommet simplicial pour G, ce qui conclut la démonstration. □

Lemme 3 ([FG69]). *Un graphe est triangulé ssi il possède un ordre d'élimination simpliciale*[3].

Démonstration.
⇒ : cette implication découle de l'application récursive du lemme précédent : dans un graphe triangulé, on trouve un sommet simplicial, on le supprime, le graphe obtenu reste un graphe triangulé...
⇐ : si un graphe n'est pas triangulé, alors il possède un cycle de plus de 3 sommets sans corde, et aucun sommet de ce cycle ne peut être éliminé "simplicialement". □

Théorème 15 (Caractérisation LexBFS [RTL76]). *Un graphe G est triangulé ssi tout ordre LexBFS de G est simplicial.*

D'autres caractérisations des graphes triangulés sont proposées dans le théorème 17.

5.4.2 Reconnaissance des graphes d'intervalles

5.4.3 Historique des algorithmes

On rappelle tout d'abord la définition d'un graphe d'intervalles.

Définition 17. *Un graphe $G = (V = \{v_1, \dots, v_n\}, E)$ est un **graphe d'intervalles** s'il est graphe d'intersection d'un ensemble d'intervalles de réels, c'est à dire qu'il existe un ensemble d'intervalles $\{I_i \subset \mathbb{R}\}$, $1 \leqslant i \leqslant n$, appelé **réaliseur** (ou réalisation) de G tel que :*

$$\forall i \neq j \in [\![1, n]\!], (v_i, v_j) \in E \Leftrightarrow I_i \cap I_j \neq \emptyset.$$

Il existe plusieurs caractérisations des graphes d'intervalles, dont certaines sont présentées dans le théorème 18.

Un historique non exhaustif des algorithmes de reconnaissance des graphes d'intervalles est présenté en figure 5.6.

5.4.4 Approche en deux étapes

De nombreuses classes de graphes admettent des caractérisations par un ordre sur les sommets du graphe. L'algorithme de reconnaissance est alors de la forme :
– calculer un ordre,
– vérifier l'ordre trouvé.

[3] L'ordre total σ sur un graphe G est un **ordre d'élimination simpliciale** de G si pour tout $i \in [\![1, n]\!]$, $\sigma(i)$ est un sommet simplicial de $G[\{\sigma(i), \sigma(i+1), \dots, \sigma(n)\}]$, c'est à dire qu'on peut retirer tous les sommets de G progressivement dans l'ordre σ tel que tout sommet enlevé ait pour voisinage une clique.

Auteurs, références	Complexité	Commentaires
Lekkerkerker, Boland [LB62]	$O(n^4)$	sommets simpliciaux et triplets astéroïdes
Fulkerson, Gross [FG69]	$O(n^4)$	1-consécutifs pour matrice des cliques max
Booth, Lueker [BL75, BL76]	$O(n+m)$	utilisation de PQ-trees
Korte, Mohring [KM86, KM89]	$O(n+m)$	utilisation de MPQ-trees et LexBFS
Ramalingam, Pandu Rangan [RR90]	$O(n^2)$	algo séquentiel et parallèle
Hsu, Ma [HM91, HM99]	$O(n+m)$	utilisation de décomposition modulaire et LexBFS
Simon [Sim92]	$O(n+m)$	4 LexBFS successifs, *contrexemple trouvé par Ma*
Hsu [Hsu93]	$O(n+m)$	variante de [HM91] pour la déc. modulaire
Corneil, Olariu, Stewart [COS98]	$O(n+m)$	4 LexBFS successifs
Habib, Paul, McConnell, Viennot [HPMV00]	$O(n+m)$	LexBFS et affinage de partition sur les cliques max.

FIG. 5.6 – Historique de la reconnaissance des graphes d'intervalles.

Les deux étapes peuvent avoir une complexité différente.

On a déjà vu dans le lemme 3 que pour reconnaître un graphe triangulé on peut chercher un ordre d'élimination simpliciale de ses sommets. De même, reconnaître un cograph revient à chercher un ordre d'élimination par sommets jumeaux. Pour les graphes de comparabilité, on peut trouver un ordre sur les sommets qui est une extension linéaire[4] de l'ordre associé au graphe.

On a aussi une caractérisation des graphes de co-comparabilité et des graphes d'intervalles par un ordre avec une propriété particulière. De plus il existe dans les deux cas un ordre LexBFS satisfiant cette propriété.

Propriété 10 (Caractérisation des graphes de co-comparabilité)**.** *G est un graphe de co-comparabilité ssi il existe un ordre σ sur ses sommets tel que $\forall x \preceq_\sigma y \preceq_\sigma z$, si $(x,z) \in E$ alors $(x,y) \in E$ ou $(y,z) \in E$.*

Propriété 11 (Caractérisation des graphes d'intervalles [Ola91])**.** *G est un graphe d'intervalles ssi il existe un ordre σ sur ses sommets tel que $\forall x \preceq_\sigma y \preceq_\sigma z$, si $(x,z) \in E$ alors $(x,y) \in E$ (voir figure 5.7).*

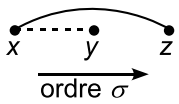


FIG. 5.7 – x, y et z forment une ombrelle pour l'ordre σ : tout graphe d'intervalles admet un **ordre sans ombrelle** sur ses sommets : par exemple, en considérant un réaliseur du graphe, l'ordre des extrémités gauches, en allant de gauche à droite.

On a une propriété similaire pour les graphes de permutation[5] :

Propriété 12 (Caractérisation des graphes de permutation)**.** *G est un graphe de permutation ssi il existe un ordre σ sur ses sommets tel que $\forall x \preceq_\sigma y \preceq_\sigma z$:*
- *si $(x,z) \in E$ alors $(x,y) \in E$ ou $(y,z) \in E$,*
- *si $(x,z) \notin E$ alors $(x,y) \notin E$ ou $(y,z) \notin E$.*

5.4.5 Approche par la chaîne des cliques maximales

Algorithme 3 (Reconnaissance des graphes d'intervalles [HPMV00])**.**

[4] extension linéaire : voir chapitre 7.2.
[5] graphe de permutation : voir chapitre 7.4

– *Calcul à l'aide d'un parcours LexBFS d'un arbre des cliques maximales (il y en a $O(m+n)$), aux arcs orientés depuis les dernières vers les premières cliques trouvées.*
– *Affinage de partition en utilisant les séparateurs maximaux et en commençant par la dernière clique trouvée dans l'arbre appelée C_1 : la chaîne de cliques recherchée est de la forme $\big(C_1$ (les cliques qui contiennent les éléments du séparateur maximal entre C_1 et sa clique "mère" dans l'arbre des cliques) (les autres cliques)$\big)$.*
– *Vérification finale : chaque sommet doit se trouver dans des cliques consécutives au sein de la chaîne de cliques. Sinon, le graphe n'est pas un graphe d'intervalles.*

5.4.6 Approche par les sous-graphes exclus

Il existe aussi une caractérisation des graphes d'intervalles par sous-graphes exclus, qu'on peut illustrer par le problème du Duc de Densmore.

Exercice 6 (Le problème du Duc de Densmore, d'après Claude Berge [Ber94]).
Le Duc de Densmore est retrouvé mort dans l'explosion d'une pièce de son château de l'île de White. Le meurtre a été commis à l'aide d'une bombe placée avec précaution dans le labyrinthe du château, ce qui a nécessité une longue préparation en cachette dans le labyrinthe. Or, avant son assassinat, le duc avait invité 8 femmes sur l'île, celles-ci se rappellent quelles autres femmes elles y ont vu, mais ont oublié à quelle date précise elles y étaient :
– Ann a vu Betty, Cynthia, Emily, Felicia et Georgia ;
– Betty a vu Ann, Cynthia et Helen ;
– Cynthia a vu Ann, Betty, Diana, Emily et Helen ;
– Diana a vu Cynthia et Emily ;
– Emily a vu Ann, Cynthia, Diana et Felicia ;
– Felicia a vu Ann et Emily ;
– Georgia a vu Ann et Helen ;
– Helen a vu Betty, Cynthia et Georgia.
Le marin qui faisait la navette vers l'île a aussi oublié les dates, mais il se souvient n'avoir transporté chacune que pour un seul aller retour. Le graphe de la relation "a vu", représenté en figure 5.8 doit donc être un graphe d'intervalles (chaque sommet représentant l'intervalle de temps qu'une invitée a passé sur l'île), donc tout sous-graphe induit doit être un graphe d'intervalles.

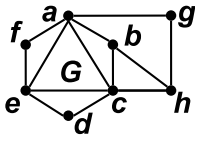


FIG. 5.8 – Le graphe ci-dessus, dont les sommets sont associés aux invitées sur du Duc de Densmore devrait être un graphe d'intervalles si chaque invitée était restée pendant un intervalle de temps continu sur l'île, ce qui l'aurait empêché de s'absenter pour préparer l'attentat.

Or les 3 sous-graphes suivants posent problème :
– $G[\{a,b,g,h\}]$ (les intervalles correspondant à a et h sont disjoints, donc ceux de b et g doivent se trouver entre les deux pour les intersecter, or ils sont aussi disjoints : impossible),
– $G[\{a,c,g,h\}]$ (même raisonnement : C_4 n'est pas un graphe d'intervalles),
– $G[\{a,b,c,d,e,f\}]$ (les intervalles correspondant à f, b, et d doivent être disjoints, considérons, les autres cas se traitant de façon analogue, qu'ils sont placés dans cet ordre ; alors celui correspondant à e doit se trouver entre les intervalles de f et d, ce qui est impossible puisqu'il n'intersecte pas l'intervalle de b).

On remarque que le noeud a est le seul à apparaître dans ces trois sous-graphes interdits : Ann est donc certainement la coupable !

Chapitre 6

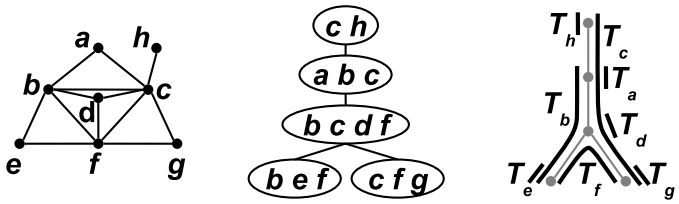
Largeur arborescente

Ce concept est à l'intersection de nombreux domaines (satisfaction de containtes, bases de données, optimisation). On a en particulier des résultats très divers utilisant des propriétés des graphes à largeur arborescente bornée.

On pourra aussi se reporter à la synthèse de Bodlaender [Bod05], ou à sa page web sur le sujet : http://www.cs.uu.nl/~hansb/treewidthlib/.

6.1 Arbre de cliques

Définition 18. *Un **arbre de cliques maximales** d'un graphe triangulé G est un arbre T dont les nœuds sont les cliques maximales, les arêtes sont les séparateurs minimaux, et sur lequel chaque sommet de G correspond à un sous-arbre (induit par l'ensemble des noeuds de l'arbre étiquetés par une clique max qui contient ce sommet).*



(a) (b) (c)

FIG. 6.1 – Un graphe triangulé G (a), son arbre de cliques maximales (b), et la famille de sous-arbres de cet arbre, dont G est le graphe d'intersection (c). Chaque arête de l'arbre de cliques maximales peut être étiquetée par un séparateur minimal de G, l'ensemble des sommets communs aux deux cliques reliées par l'arête.

Un tel arbre peut être obtenu par l'algorithme LexBFS [RTL76].

Théorème 16 (Caractérisation des graphes d'intervalles [GH64])**.** *Un graphe est un graphe d'intervalles ssi il admet un arbre des cliques qui est une chaîne.*

Voir l'algorithme 3.

Théorème 17 (Caractérisation des graphes triangulés [Dir61, FG69, Gav74])**.** *Soit G un graphe. Les quatre propositions suivantes sont équivalentes :*

(i) G triangulé.

(ii) G admet un schéma d'élimination simpliciale.

(iii) G admet un arbre de cliques maximales.

(iv) G est le graphe d'intersection d'une famille de sous-arbres d'un arbre.

Démonstration. $(i) \Leftrightarrow (ii)$: voir lemme 3.

$(iii) \Rightarrow (iv)$: G est le graphe d'intersection d'une famille (T_v) de sous-arbres de l'arbre de cliques maximales, chaque sous-arbre T_v recouvrant les nœuds de l'arbre dont l'étiquette contient le noeud v, comme montré en figures 6.1(b) et (c). □

Propriété 13. *On peut trouver une clique max d'un graphe triangulé en temps linéaire : parmi tous les sommets dans le schéma d'élimination simpliciale, on choisit le sommet v qui maximise $1 + |\mathcal{N}(v)|$. Lui et son voisinage constituent une clique max.*

Démonstration. Considérons les cliques max, et le premier sommet de ces cliques max atteint par le schéma d'élimination simpliciale. Lui même et son voisinage constituent bien la clique max qui sera trouvée par l'algorithme. □

Cet algorithme fournit aussi un algorithme de coloration : on commence par colorier la clique max, puis on effectue un coloriage glouton.

On remarque aussi qu'il y a au plus n cliques maximales dans un graphe triangulé, (alors qu'il y peut y en avoir 2^n dans un graphe quelconque). On en déduit que l'arborescence des cliques maximales se calcule en $O(n + m)$.

En revanche, le graphe d'intersection des cliques maximales se calcule en $O(n^2)$[1].

Théorème 18 (Caractérisation des graphes d'intervalles)**.** *Soit G un graphe. Les quatre propositions suivantes sont équivalentes :*

(i) G graphe d'intervalles.

(ii) G triangulé et $\overline{G}$ graphe de comparabilité.

(iii) G admet une chaîne de cliques maximales, c'est-à-dire qu'il existe un ordre total sur les cliques maximales tel que pour tout v les cliques contenant v sont consécutives.

Démonstration. $(i) \Rightarrow (ii)$: si on connaît une réalisation de G, en traçant le graphe de la relation "est à gauche de" pour les intervalles de cette réalisation, on obtient bien $\overline{G}$ (on en a de plus une orientation transitive). □

6.2 k-arbres

Définition 19. *Un k-**arbre** se construit par récurrence de la façon suivante :*

- *une clique à $k + 1$ sommets est un k-arbre.*
- *si G est un k-arbre, en ajoutant un nouveau sommet adjacent à exactement l'ensemble des sommets d'une k-clique de G, on obtient un k-arbre G'.*

Remarque 1. Les 1-arbres sont les arbres. Les 2-arbres (voir un exemple en figure 6.2) sont des graphes séries-parallèles (voir propriété 17).

[1] exemple pour indiquer que ça ne peut se calculer en $O(n + m)$?

Remarque 2. Tout k-arbre est un graphe triangulé, et toutes ses cliques maximales ont $k+1$ sommets.

Propriété 14 ([Ros74])**.** *Un graphe G est un k-arbre ssi il est connexe, qu'il a une k-clique et pas de $k+2$-clique, et que tout séparateur minimal est une k-clique.*

Propriété 15. *Un graphe G à strictement plus de k sommets est un k-arbre ssi il existe un schéma d'élimination $(v_1, v_2, \ldots, v_n)$ tel que chaque sommet v_i, pour $i \in [\![1, n-k]\!]$, est adjacent à une k-clique dans $G[v_i, v_{i+1}, \ldots, v_n]$.*

Propriété 16. *Si G est un k-arbre alors il est triangulé et toutes ses cliques maximales ont $k+1$ sommets.*

Définition 20. *Un k-**arbre partiel** est un graphe partiel d'un k-arbre (c'est-à-dire un k-arbre duquel on a enlevé des arêtes).*

Propriété 17. *La classe des graphes séries-parallèles[2] est exactement celle des 2-arbres partiels.*

Définition 21. *La **largeur arborescente** d'un graphe G est le plus petit k tel que G est un k-arbre partiel.*

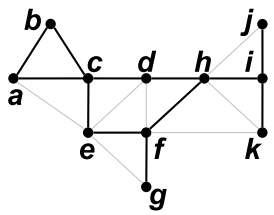


FIG. 6.2 – Graphe de largeur arborescente 2 : en le triangulant, c'est à dire en ajoutant les fines arêtes grises, on obtient un 2-arbre.

L'idée de cette définition est donc qu'on va compléter G pour en faire un graphe triangulé "spécial", un k-arbre, comme illustré en figure 6.2. On va donc trianguler G, ce qui motive la définition suivante :

Définition 22. *Une **complétion triangulée** ou **triangulation** d'un graphe $G = (V, E)$ est un graphe $H = (V, F)$ tel que $E \subseteq F$ et H est un graphe triangulé.*

On a une définition similaire pour les graphes d'intervalles :

Définition 23. *Une **complétion d'intervalles** d'un graphe $G = (V, E)$ est un graphe $H = (V, F)$ tel que $E \subseteq F$ et H graphe est un graphe d'intervalles.*

Ces définitions permettent de définir quatre problèmes, prenant en entrée un graphe G :

[2] un **graphe série-parallèle** se définit par récurrence de la façon suivante : une paire de sommets adjacents est un graphe série-parallèle, tout comme la composition série et la composition parallèle de deux graphes séries-parallèles. La composition parallèle de deux graphes G et G' est définie par rapport à deux sommets x et y de G et x' et y' de G' de la façon suivante : on identifie les sommets x et x' d'une part, y et y' d'autre part puis on fait l'union des arêtes et des sommets des deux graphes. Pour la composition série, on effectue le même processus, en identifiant non pas les sommets x avec x' et y avec y', mais en identifiant y et x'. En d'autres termes, on choisit une source et une destination à chacun des deux graphes qu'on considère comme de simples segments pour soit les recoller en parallèle, soit les recoller l'un derrière l'autre.

Problème 1 (MinimumFill-In, ChordalCompletion). *Trouver une triangulation $H = (V, F)$ de $G = (V, E)$ minimisant $|F \backslash E|$, le nombre d'arêtes ajoutées.*

Théorème 19 ([Yan81]). *Le problème* MinimumFill-In *est NP-complet.*

Problème 2 (Treewidth). *Trouver une triangulation H de G ayant une largeur arborescente minimale, c'est-à-dire une clique maximum de taille minimale.*

Théorème 20 ([ACP87]). *Le problème* Treewidth *est NP-complet.*

Problème 3 (IntervalCompletion). *Trouver une complétion d'intervalles $H = (V, F)$ de $G = (V, E)$ minimisant $|F \backslash E|$, le nombre d'arêtes ajoutées.*

Théorème 21 ([KF79a, Yan81], [GJ79] GT35). *Le problème* IntervalCompletion *est NP-complet.*

Problème 4 (Pathwidth). *Trouver une complétion d'intervalles H de G ayant une clique maximum de taille minimale (ce qui correspond à une largeur linéaire, concept défini dans la section suivante, minimale).*

Théorème 22 ([KF79b, ACP87]). *Le problème* Pathwidth *est NP-complet.*

Même si ces problèmes sont NP-complets, et le restent sur des classes de graphes même très restreintes (les graphes bipartis par exemple [Klo93]), il existe pour les résoudre des algorithmes d'approximation [Bod05].

6.3 Décompositions arborées

Cette approche correspond à celle utilisée par Robertson et Seymour pour introduire les notions de largeur arborescente et largeur linéaire [RS83, RS86].

Définition 24. *Une **décomposition arborescente** d'un graphe $G = (V, E)$ est une paire $D_T(G) = (S, T)$, où $S = \{V_i / i \in I\}$ est une collection de sous-ensembles de V, et T un arbre aux nœuds étiquetés par les éléments de S tel que :*

(i) $\bigcup_{i \in I} V_i = V$,

(ii) $\forall e \in E, \exists i \in I$ *tq* $e \in E \cap V_i^2$ *(pour toute arête de E, un des sous-ensembles V_i contient les deux sommets joints par cette arête),*

(iii) $\forall v \in V$, *les V_i contenant v forment un sous-arbre de T.*

Définition 25. *Une **décomposition de chemins** d'un graphe $G = (V, E)$ se définit aussi comme une paire $D_C(G) = (S, T)$ avec les mêmes propriétés, mais de plus T est un chemin.*

En utilisant ces décompositions illustrées en figure 6.3, on peut obtenir une définition équivalente de la largeur arborescente et de la largeur linéaire.

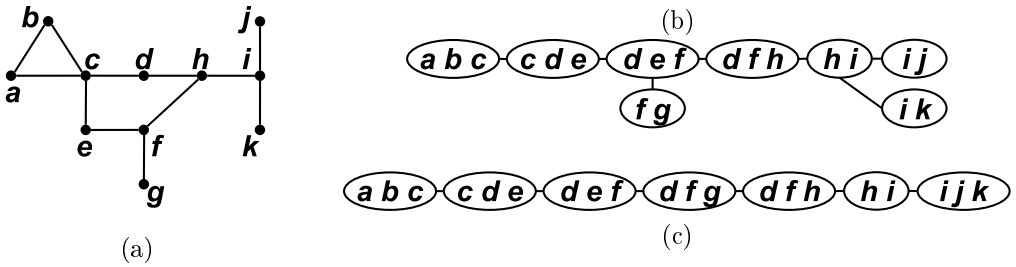


FIG. 6.3 – Décompositions arborées d'un graphe (a) : une décomposition arborescente (b) et une décomposition de chemins (c).

Théorème 23. *La **largeur arborescente** d'un graphe G, notée $tw(G)$ est le minimum, parmi toutes les décompositions arborescentes de G, de la taille de leur plus grande étiquette -1 :*

$$tw(G) = \min_{D_T(G)=(S,T)} \{\max_{V_i \in S} |V_i| - 1\}$$

*La **largeur linéaire** d'un graphe G, notée $pw(G)$ est le minimum, parmi toutes les décompositions de chemins de G, de la taille de leur plus grande étiquette -1 :*

$$pw(G) = \min_{D_C(G)=(S,T)} \{\max_{V_i \in S} |V_i| - 1\}$$

Démonstration. On obtient facilement les inégalités dans les deux sens. □

Propriété 18. *La largeur arborescente des graphes bipartis complets, des grilles, et des graphes complets est connue :*

(i) $tw(K_{i,j}) = \min(i, j)$.

(ii) $tw(G_{i\times j}) = \min(i, j)$ *($G_{i\times j}$ étant la **grille** d'ordre i, j.)*

(iii) $tw(K_n) = n - 1$.

Démonstration. *(i)* Soit $K_{i,j}$ un graphe biparti complet, dont l'ensemble des sommets se partitionne en V_1 de taille i, et V_2 de taille j. Soient a et b deux sommets de V_1, c et d deux sommets de V_2. Dans la triangulation de $K_{i,j}$ correspondant à un $tw(K_{i,j})$-arbre, supposons que a et b ne sont pas adjacents, et c et d non plus. Alors comme on a $c - a$ et $c - b$ et $d - a$ et $d - b$ alors on a un cycle C_4 sans corde : absurde ! Donc soit a et b sont reliés, soit c et d le sont, donc $tw(K_{i,j}) \geqslant \min i, j$.

Trivialement cette borne est atteinte (on triangularise en reliant deux à deux tous les sommets du plus petit ensemble parmi V_1 et V_2), donc on a bien l'égalité voulue. □

Ces décompositions permettent d'appliquer des algorithmes de programmation dynamique sur l'arbre de décomposition arborescente, enraciné sur un sommet. L'application de tels algorithmes prendra donc du temps sur les feuilles contenant le plus de sommets, ce qui explique qu'on préfère travailler sur des graphes de largeur arborescente bornée.

De tels graphes apparaissent dans de nombreux domaines :

- le graphe de flux de contrôle de programmes sans instruction GOTO dans un certain nombre de langages impératifs (comme C, Pascal...) a une largeur arborescente bornée par une petite constante [Tho98]. Ces résultats expliquent pourquoi les programmes de coloration utilisés pour affecter les registres lors de la compilation fonctionnaient si bien.
- problème du voyageur de commerce (qui consiste à trouver le cycle hamiltonien de poids le plus faible dans un graphe complet aux arêtes valuées) : Chvátal propose de calculer cinq cycles hamiltoniens en utilisant cinq heuristiques différentes de résolution du problème du

voyageur de commerce, et considérer le graphe formé par la superposition de ces cycles, qui est un graphe de largeur arborescente bornée. Il est donc facile de calculer dessus un cycle hamiltonien de poids minimal, qui sera une excellente initialisation pour un programme de séparation et évaluation.
- satisfaction de contraintes [Fre90, DKV02].
- routage et étiquetage de distance sur les réseaux [BTTvL97].
- bases de données : hypergraphes acycliques [GLS01].

Théorème 24 (Le voleur et les gendarmes [ST93b]). *Soit un graphe G non orienté. Un voleur et k gendarmes se trouvent sur des sommets du graphe. Le voleur peut à tout instant courir en utilisant les arêtes du graphes vers tout autre sommet. Il ne peut toutefois passer par un sommet occupé par un gendarme. Les gendarmes peuvent eux passer d'un sommet à un autre quelconque (sans nécessairement utiliser les arêtes du graphes, disons qu'ils sont sur des hélicoptères), ils connaissent à tout instant la position du voleur, et cherchent à le cerner, occuper tous les sommets voisins de sa position. Si G a pour largeur arborescente $k-1$, alors $k-1$ gendarmes ne suffisent pas à le cerner, mais k suffisent. De plus ils peuvent utiliser une stratégie monotone, c'est à dire parcourir progressivement le graphe en assurant que le voleur ne peut retourner dans les parties parcourues*

Théorème 25 ([CMR98]). *Sur les graphes de largeur arborescente bornée, tout problème de décision ou d'optimisation exprimable en logique monadique du second ordre peut être résolu par un algorithme linéaire.*

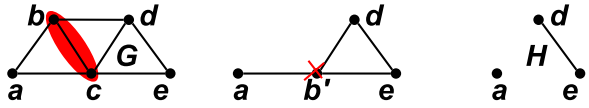
Cette bonne nouvelle est tout de même nuancée par le fait que la constante est énorme, c'est une tour d'exponentielles de la forme $\left(2^{2^{2^{\cdots^2}}}\right)^k$ dont la hauteur dépend de la formule du second ordre et n'est pas bornée (à moins que P=NP) [FG04].

6.4 Mineurs

En complément de ce chapitre, on pourra utiliser une synthèse de László Lovász sur la théorie des mineurs de graphes [Lov05].

Définition 26. *Un graphe H est un mineur de G s'il peut être obtenu par des contractations d'arêtes d'un sous-graphe induit de G, ou à partir de G en effectuant un nombre quelconque d'opérations parmi les suivantes :*
- *suppression d'une arête,*
- *contraction d'une arête,*
- *suppression d'un sommet et de toutes les arêtes adjacentes.*

On le note $H \trianglelefteq G$ (illustré en figure 6.4).



(a) (b) (c)

FIG. 6.4 – Un graphe G (a) et son mineur H obtenu par contraction d'arête (b) puis suppression de sommet (c).

Remarquons que $\trianglelefteq$ est une relation d'ordre sur les graphes.

Théorème 26 (SR1). *Soit $\mathcal{G}$ une classe de graphes close par mineur, alors il existe un ensemble fini d'obstructions $ob(\mathcal{G})$ tq $G \in \mathcal{G}$ ssi $\nexists H \in ob(\mathcal{G})$ tq H mineur de G.*

Exemple : la classe des graphes planaires étant close par mineur, il existe un ensemble d'obstructions $\{K_5, K_{3,3}\}$, tel qu'aucune de ces deux obstructions n'est mineur d'un graphe planaire.

Théorème 27 (SR2). *Soit un graphe G à n sommets. Pour tout graphe H, il existe un algo en $O(n^3)$ qui teste si H est mineur de G.*

Propriété 19. *Si G mineur de H alors $tw(G) \leqslant tw(H)$.*

De plus, les graphes de largeur arborescente bornée sont caractérisables en terme de mineurs : ceux de largeur arborescente inférieure ou égale à 2 par exemple sont les graphes n'ayant pas K_4 comme mineur.

Chapitre 7

Dualité ordres graphes

Les éléments de ce chapitre sont aussi abordés dans des notes de cours de Brightwell et Trotter [BT06].

7.1 Introduction : les ordres d'intervalles

A tout graphe d'intervalles on peut associer un ensemble d'intervalles ordonnés selon l'ordre $\preceq$ défini comme suit : $I_1 \preceq I_2$ ssi I_1 est totalement à gauche de I_2. Si au contraire I_1 et I_2 ont une intersection non vide alors ils sont dits **incomparables**.

Ainsi, pour un ensemble d'intervalles de réels, son graphe d'intervalles est le complémentaire du graphe de comparabilité pour l'ordre "totalement à gauche de". Plus précisément, soit G un graphe d'intervalles et soit P un ordre d'intervalles défini sur le même ensemble d'intervalles, alors P est une orientation transitive sans circuit du complémentaire de G. On remarque au passage qu'il est possible d'associer plusieurs ordres d'intervalles à un même graphe puisque l'orientation transitive n'est pas unique.

Cette dualité se rencontre aussi pour d'autres classes de graphes[1] :
- ordres de dimension 2 : graphes de permutation,
- ordres séries-parallèles : cographes.

On évoquera notamment dans ce chapitre les classes de la figure 7.1.

7.2 Définitions de base sur les ordres

Plusieurs définitions des ordres partiels se trouvent dans la littérature. Historiquement, dans les articles de Dushnik et Miller [DM41] et le livre de Fishburn [Fis85]

Définition 27. *Une **relation binaire** R sur un ensemble S est un sous-ensemble de S^2 (remarquons alors que toute relation peut être trivialement codée par un multigraphe contenant éventuellement des boucles). Pour exprimer que $(a, b) \in R$, on peut noter $a \preceq_R b$. Une relation binaire P sur un ensemble S est un ordre partiel si elle est **irréflexive** ($\forall x \in S, x \not\preceq_P x$) et **transitive** ($\forall x, y, z \in S, x \preceq_P y$ et $y \preceq_P z \Rightarrow x \preceq_P z$). La paire (S, P) est alors appelée **ensemble partiellement ordonné**. Si $\forall x, y \in S$, soit $x \preceq_P y$, soit $y \preceq_P x$, alors $\preceq_P$ est un **ordre total**.*

Cette définition est compatible, dans notre étude en relation avec les graphes, avec celle définissant une relation d'ordre P comme une relation binaire réflexive, antisymétrique, et transitive, sous réserve de ne pas noter les boucles dans le graphe, et de n'utiliser qu'une arête non orientée pour exprimer à la fois que $x \preceq_P y$ et $y \preceq_P x$. La réflexivité pourra de plus être utile en pratique pour certaines relations d'ordre (l'inclusion non stricte par exemple).

[1] Site sur les classes de graphes : http://wwwteo.informatik.uni-rostock.de/isgci/

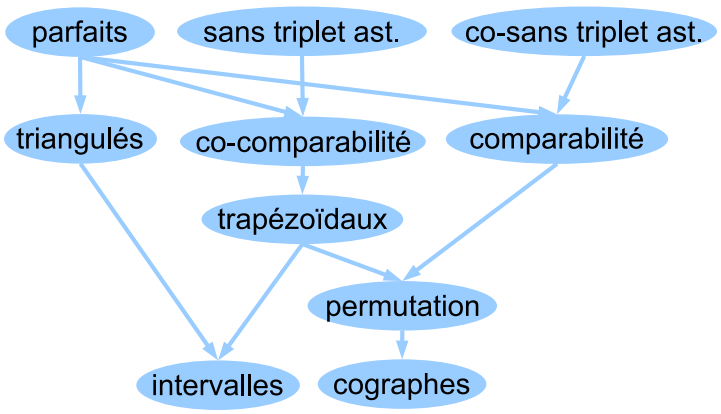


FIG. 7.1 – Réduction transitive du graphe d'inclusion de classes de graphes : un arc de la classe A vers la classe B indique que $B \subset A$, et les arcs que l'on peut déduire par transitivité ont été effacés. La classe des graphes d'intervalles est égale à la classe des graphes sans triplets astéroïdes et triangulés. Celle des graphes de permutation est égale à la classe des graphes de comparabilité et de **co-comparabilité** (c'est-à-dire les graphes dont le complémentaire est un graphe de comparabilité).

Définition 28. *La **couverture** d'une relation d'ordre P est notée $\prec$ et définie comme suit : $\forall x, y \in S,\ x \prec_P y \Leftrightarrow \forall z \in S, x \preceq_P z \preceq_P y \Rightarrow z = x$ ou $z = y$ (x est le prédécesseur immédiat de y par la relation $\preceq$).*
*Une **antichaîne** est un ensemble de sommets deux à deux **incomparables** (on le note généralement $x \,||_P\, y$, ce qui signifie que ni $x \preceq_P y$ ni $y \preceq_P x$), c'est donc un ensemble stable dans le graphe de comparabilité. Inversement, une **chaîne** est un ensemble de sommets tous comparables deux à deux, c'est une clique dans le graphe de comparabilité.*
*Une **extension linéaire** L d'un ordre P est un ordre total compatible avec l'ordre P ($\forall x, y \in S, x \preceq_P y \Rightarrow x \preceq_L y$).*
*La **fermeture transitive** $(\preceq_P)^t$ d'une relation binaire $\preceq_P$ est définie par : $\forall x, y \in S, x \preceq_P y \Rightarrow x\,(\preceq_P)^t\,y$ et $\forall x, y, z \in S, x\,(\preceq_P)^t\,y$ et $y\,(\preceq_P)^t\,z \Rightarrow x\,(\preceq_P)^t\,z$. La **fermeture transitive** d'un graphe orienté (pour un graphe non orienté, considérer une orientation de ses arêtes permet d'en obtenir une fermeture transitive) est définie de façon similaire (si on peut effectuer une marche d'un sommet x à un sommet y on ajoute l'arête (x, y)).*
*Une **réduction transitive** d'un graphe G non orienté est un graphe de même clôture transitive que G ayant un nombre d'arêtes minimal. Un graphe orienté sans circuit a une unique réduction transitive appelée **diagramme de Hasse**.*

On peut donc représenter un ordre P par divers graphes :
– le graphe de fermeture transitive $G^t(P)$ (graphe de comparabilité),
– le graphe de réduction transitive $G_r(P)$ (diagramme de Hasse, graphe de couverture),
– tout graphe G tel que $G^t = G^t(P)$ (qui est un graphe sandwich de $G_r(P)$ et $G^t(P)$)

Remarquons qu'il est possible d'avoir un diagramme de Hasse avec $O(n)$ arêtes correspondant à un graphe de comparabilité à $O(n^2)$ arêtes (pour l'ordre total sur n éléments par exemple, respectivement la chaîne à $n-1$ arêtes et la clique à $\frac{n(n-1)}{2}$ arêtes).

La réduction et la fermeture transitive se calculent en en $O(mn)$ (ou en la complexité du meilleur algorithme de multiplication de matrices). Le faire plus efficacement est un problème ouvert.

Théorème 28 ([NR87]). *Déterminer si un graphe est un graphe non orienté de couverture est NP-complet.*

Démonstration. La preuve originale de Nešetřil et Rödl [NR87] est buggée, réparée par ses auteurs [NR93], puis réécrite en détails [NR95], Brightwell proposant une autre preuve plus simple [Bri93]. □

7.3 Dimension d'un ordre

Le lemme suivant permet d'obtenir une extension linéaire de tout ordre partiel (le résultat est difficile à démontrer plutôt pour les ordres infinis).

Lemme 4 ([Szp30]). *Soit un ensemble partiellement ordonné (S, P) et $a \,||_P\, b \in S$ alors l'ordre $\big(P \cup \{(a,b)\}\big)^t$ (obtenu en ajoutant la relation $a \preceq_P b$ et en en prenant la clôture transitive) est un ordre partiel.*

Théorème 29 ([DM41]). *Soit un ensemble partiellement ordonné (S, P). Alors il existe une famille d'extensions linéaires de P, $(L_i)_{i\in[\![1,k]\!]}$, qui réalise P : $P = \bigcap_{i\in[\![1,k]\!]} L_i$ (c'est-à-dire $\forall x, y \in S$, $x \preceq_P y \Leftrightarrow \forall i \in [\![1,k]\!], x \preceq_{L_i} y$)*

Choisir une telle famille d'extensions linéaires de taille minimale permet de définir la dimension d'un ordre.

Définition 29. *Soit P un ordre partiel, la **dimension** de P est :*

$$dim(P) = \min\left\{ |\{L_i\}| \text{ tel que } L_i \text{ extensions linéaires de } P \text{ et } P = \bigcap_{i\in[\![1,|\{L_i\}|]\!]} L_i \right\}$$

Exemple : l'ordre $P = \{(2,1),(4,1),(4,3)\}$ est l'intersection de deux extensions linéaires $L_1 = 4321$ et $L_2 = 2413$, et il est de dimension 2. Mais il n'est pas l'intersection de L_2 et $L_3 = 4231$: on a $2 \preceq 3$ dans ces deux extensions linéaires, dont l'intersection est donc $\big(P \cup \{(2 \preceq 3)\}\big)^t$.

Théorème 30 ([Yan81]). *Déterminer si un ordre est de dimension k, pour $k \geqslant 3$, est NP-complet.*

Démonstration. Par réduction de 3-Coloration, qui est NP-complet [GJS76, GJ79](GT4). □

En revanche on sait reconnaître en temps polynomial les ordres de dimension 1 (les ordres totaux) et ceux de dimension 2 dont les graphes de comparabilité sont les graphes de permutation, comme détaillé.

Théorème 31 ([BW91]). *Calculer le nombre d'extensions linéaires d'un ordre est #P-complet.*

7.4 Graphes de permutation

Soit $\pi = \binom{1 \quad \dots \quad n}{\pi(1) \quad \dots \quad \pi(n)}$ une permutation de $\mathcal{S}_n$. Le graphe formé des sommets $v_i, i \in \{1,\dots,n\}$ reliés ssi $(i-j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0$ (i et j sont inversés par la permutation) est un **graphe de permutation**.

Un graphe de permutation a diverses représentations montrées en figure 7.2 pour le graphe de la permutation $\binom{1234}{2413}$.

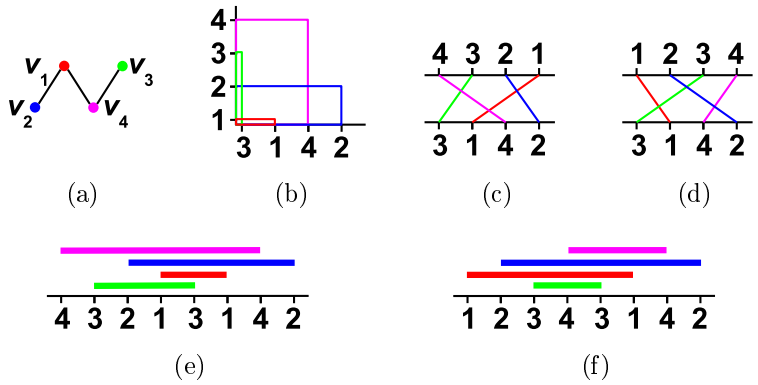


FIG. 7.2 – Un graphe de permutation (a) (dont l'ordre correspondant est l'intersection de 4321 et 2413) peut être représenté en tant que graphe d'inclusion de rectangles (b), graphe d'intersection de segments (c), graphe de précédence de segments (d), graphe d'inclusion d'intervalles (e) et graphe de chevauchement d'intervalles (f).

La représentation en graphe d'inclusion de rectangles en particulier fait apparaître qu'un ordre de permutation est un ordre de dimension 2. En effet, l'ordre $\{(2,1),(4,1),(4,3)\}$, dont le graphe de comparabilité est représenté en figure 7.2(a) est de dimension 2, ordre produit de ses extensions linéaires $L_1 = 4321$ et $L_2 = 2413$. Plus généralement tout ordre partiel P est un ordre produit qu'on peut plonger dans $\mathbb{N}^{dim(P)}$.

Théorème 32 ([DM41]). *Soit un graphe G. On a les équivalences suivantes :*
– G est un graphe de permutation,
– G admet une orientation transitive qui correspond à un ordre de dimension 2,
– G et $\overline{G}$ sont des graphes de comparabilité.

Remarquons que le complémentaire d'un graphe de la permutation $\pi \in \mathcal{S}_n$ correspond à la permutation inverse $i \mapsto \pi(n-i)$.

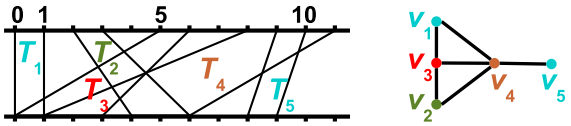
Définition 30. *Soient P et Q deux ordres ayant le même graphe de comparabilité, et S un ensemble. On appelle* ***invariant de comparabilité*** *une fonction $f : \{ordres\} \to S$ telle que si P et Q ont même graphe de comparabilité alors $f(P) = f(Q)$.*

Par exemple, avec $S = \mathbb{N}$, le nombre de sommets du graphe de comparabilité et la dimension sont des invariants de comparabilité.

7.5 Graphes trapézoïdaux

On peut définir une extension de la classe des graphes de permutation en considérant le graphe d'intersection non plus de segments, mais de trapèzes, entre deux droites parallèles.

Pour définir cette classe des **graphes trapézoïdaux**, considérons deux lignes horizontales. Un *trapèze* entre ces deux lignes est défini comme une paire de segments, un sur celle du dessus et l'autre sur celle du dessous. Un graphe est trapézoïdal s'il existe un ensemble de trapèzes correspondant aux sommets du graphe tel que deux sommets sont joints par une arête si et seulement si les trapèzes correspondants s'intersectent. Les figures 7.3(a) et (b) illustrent cette définition.



(a) (b)

FIG. 7.3 – Un ensemble de trapèzes $\mathcal{T} = \{([0,1],[0,1]),([2,3],[4,6]),([5,6],[0,3]),([8,11],[1,7]),([9,10],[8,9])\}$ (a), et le graphe trapézoïdal associé (b).

7.6 Orientation transitive et décomposition modulaire

Le chapitre suivant qui se fonde sur le livre [Gol80] et les articles [Gal67, HM79] a été donné en partiel.

On considère un graphe $G = (V, E)$ non orienté, fini, simple (i.e. sans boucle ni arête multiple). On définit la relation Γ sur E comme suit :

Pour $(a,b),(a,c) \in E$, avec $b \neq c$, $(a,b)\Gamma(a,c)$ si $(b,c) \notin E$. On notera Γ^t la fermeture transitive de la relation Γ. Γ^t partitionne donc les arêtes de G en Γ-classes.

7.6.1 Calcul des Γ-classes

Proposer un algorithme de calcul de ces Γ-classes, l'évaluer en fonction de $n = |V|$ et $m = |E|$.

Une première idée consiste à calculer le graphe $G' = (E, E_\Gamma)$ de la relation Γ, en $O(m^2)$, puis sa fermeture transitive par un parcours en profondeur depuis chaque sommet de G', ce qui se fait donc en $O(m^3)$, soit $O(n^6)$.

Mais on peut faire beaucoup mieux, en remarquant que toutes les arêtes joignant un sommet v à tout sommet d'une même composante connexe $C_{v,i}$ de $\overline{G[\mathcal{N}(v)]}$ font partie d'une même Γ-classe.

Ainsi on peut calculer les Γ-classes avec l'algorithme suivant :
On initialise une partition des arêtes $\mathcal{P} = \big\{\{e\}, e \in E\big\}$
Pour tout v de V faire :
- calculer les composantes connexes $C_{v,1}, \ldots, C_{v,k_v}$ du sous-graphe $\overline{G[\mathcal{N}(v)]}$,
- pour toute composante connexe $C_{v,i} = v_1, \ldots, v_{|C_{v,i}|}$ faire :
  - pour tout j de 2 à $|C_{v,i}|$, fusionner dans $\mathcal{P}$ la partie contenant l'arête (v, v_j) et la partie contenant (v, v_1).

En utilisant une structure de données d'Union-Find pour $\mathcal{P}$ on obtient un algorithme en $O\big(n(n+m)\big)$, soit $O(n^3)$. On peut encore améliorer l'algorithme précédent en utilisant le voisinage à distance 2 de v.

7.6.2 Modules et Γ-classes

On notera pour une Γ-classe C, $V(C)$, l'ensemble des sommets du graphe adjacents à au moins une arête de C. Montrer que si une arête d'une classe C relie deux sommets d'un même module M, alors $V(C) \subseteq M$.

Cela découle simplement de la définition de Γ.

7.6.3 Modules connexes

Montrer que tout module connexe de G est une union de $V(C)$.

Considérons un module M connexe, alors tout sommet de M est relié à une arête, qui fait partie d'une certaine classe C, et donc M contient $V(C)$ d'après la question précédente.

7.6.4 Graphes premiers

En déduire qu'un graphe premier n'admet qu'une Γ*-classe.*

On raisonne par récurrence sur le nombre de sommets du graphe.

Initialisation : la propriété est vraie pour le graphe P_4.

Hérédité : supposons que la propriété est vraie pour tout graphe premier à $n-1$ sommets ou moins, l'est-elle pour un graphe premier $G = (V, E)$ à n sommets ?

D'après le théorème 6, G contient un graphe premier à $n-1$ sommets ou un graphe premier à $n-2$ sommets. Traitons ces deux cas séparément :

Cas 1 : Soit $V' \subset V$, $|V'| = n-1$, tel que $G[V']$ premier.

Soit $v \in V \setminus V'$. Par l'hypothèse de récurrence, toutes les arêtes de V'^2 appartiennent à la même Γ-classe, il reste à montrer que les arêtes liant v à ses voisins appartiennent à cette classe.

Montrons tout d'abord qu'il existe une telle arête qui appartient à cette classe. Comme V' n'est pas un module de G, il existe $z, t \in V'$ tels que $(v, z) \in E$ et $(v, t) \notin E$. $G[V']$ étant premier, c'est un graphe connexe, donc il existe un chemin formé de sommets de V' entre z et t, et en particulier deux sommets adjacents sur ce chemin, z' et t', tels que $(v, z') \in E$ et $(v, t') \notin E$. Ainsi, (v, z'), qui appartient à $E \setminus (V'^2)$, est dans la même Γ-classe que (t', z'), c'est à dire, par hypothèse de récurrence, l'unique Γ-classe de $G[V']$.

Montrons maintenant que toutes les arêtes de $E \setminus (V'^2)$ appartiennent à la même Γ-classe. Soient (v, y') et (v, y'') deux arêtes de $E \setminus (V'^2)$. Comme $G[V']$ est premier, $\overline{G}[V']$ est connexe, donc il existe une chaîne dans $\overline{G[V']}$ de y' à y'', et donc (v, y') et (v, y'') appartiennent à la même Γ-classe.

Cas 2 : Soit $V' \subset V$, $|V'| = n-2$, tel que $G[V']$ premier.

Soit $x, y \in V \setminus V'$, supposons $(x, y) \in E$. Par le même raisonnement que dans le cas 1, on obtient que toute arête entre x et ses voisins dans V', ainsi qu'entre y et ses voisins dans V', appartiennent à la même Γ-classe.

Il reste à vérifier que (x, y) appartient à la même Γ-classe. Comme x et y ne sont pas des sommets jumeaux, alors il existe $z \in V \setminus \{x, y\}$ t.q. $(x, z) \in E$ et $(y, z) \notin E$ (ou bien $(x, z) \notin E$ et $(y, z) \in E$) donc (x, y) appartient à la même Γ-classe que (x, z) (ou bien la même Γ-classe que (y, z)), ce qui conclut la démonstration.

7.6.5 Γ-classe unique

Réciproquement que peut-on dire des modules d'un graphe connexe n'ayant qu'une Γ*-classe ?*

Comme le graphe est connexe, le cas d'un module connexe et de sommets isolés est exclu. Donc les modules éventuels sont des stables.

7.6.6 Γ-classe globale

** Montrer qu'un graphe admet au plus une* Γ*-classe C telle que $V(C) = V$.*

La preuve directe de cette propriété est difficile, mais se fait par l'absurde.

7.6.7 Connexité et connexité du complémentaire

** Montrer qu'un graphe G connexe et dont le complémentaire est connexe admet une* Γ*-classe C telle que $V(C) = V$.*

D'après le théorème de décomposition modulaire, l'arbre de décomposition d'un tel graphe admet un noeud premier à la racine. Or on a montré qu'un graphe premier n'a qu'une R-classe.

Le graphe quotient associé à la racine est premier, possède un P_4 et donc toutes les arêtes de G (et aussi de $\overline{G}$) sont dans une seule R-classe.

7.6.8 Décomposition modulaire

En déduire un algorithme de construction de l'arbre de décomposition modulaire basé sur les Γ*-classes.*

On peut imaginer l'algorithme suivant :

Une étape de la récursivité :

- Calcul des composantes connexes de G et $\overline{G}$
- Calcul des R-classes.
- Construction du sous-graphe H engendré par l'unique R-classe qui couvre tous les sommets
- Les classes de sommets (faux, vrais) jumeaux constituent la partition en modules maximaux.

7.6.9 Orientations transitives et Γ-classes

En fait ces classes ont été introduites pour étudier les orientations transitives d'un graphe de comparabilité. Lorsque $(a, b)\Gamma^t(c, d)$*, les orientations possibles de ces arêtes dans une orientation transitive du graphe sont liées. Comment engendrer les orientations transitives d'un graphe de comparabilité à l'aide des* Γ*-classes ?*

- Un graphe de comparabilité premier n'a que deux orientations transitives
- Si G n'est pas connexe, ses composantes connexes s'orientent indépendamment. Produit des orientations possibles des composantes.
- Si $\overline{G}$ n'est pas connexe, on choisit un ordre total sur ces k composantes et cela engendre une orientation transitive de G. ($k!$ orientations possibles).
- Les modules maximaux de la partition modulaire s'orientent indépendamment.

7.6.10 Orientations transitives et décomposition modulaire

Comment engendrer les orientations transitives d'un graphe de comparabilité à l'aide de l'arbre de décomposition modulaire ?

7.6.11 Calcul des Γ-classes et décomposition modulaire

Peut-on déduire un algorithme de calcul des Γ*-classes à partir de l'arbre de décomposition modulaire ? Comparer à votre réponse de la première question.*

L'algorithme obtenu est en $O(n + m)$. Moralité : un théorème de structure peut améliorer les algorithmes !

7.6.12 Reconnaissance des graphes de comparabilité

Ecrire un algorithme de reconnaissance des graphes de comparabilité basé sur les Γ*-classes.*

Tout graphe premier n'est pas de comparabilité, il suffit donc de reconnaître les graphes premiers...

Chapitre 8

Annexe

8.1 Lexique français-anglais

vertex (vertices)	sommet (sommets)
arête	edge
graphe complémentaire	complement graph
voisinage	neighbourhood
jumeaux	twins
connexe	connected
composante connexe	connected component
non connexe	disconnected
sous-graphe induit	induced subgraph
isomorphe	isomorphic
chemin	path
biparti	bipartite
chevauchant	overlapping
module fort	strong module
unicité	uniqueness
symétrie	symmetry
diviser pour régner	divide and conquer
sans perte de généralité	without loss of generality (wlog)
taureau	bull
casseur	splitter
affiner	refine
sous-modulaire	submodular
affinage de partition	partition refinement
treillis des partitions	partition lattice
parcours en largeur lexicographique	lexicographic breadth-first search
excentricité	eccentricity
diamètre	diameter
rayon	radius
graphe série-parallèle	series-parallel graph
graphe triangulé	triangulated graph, chordal graph
ordre sans ombrelles	ordering without umbrellas
largeur arborescente	treewidth
largeur (arborescente) linéaire	pathwidth
complétion triangulée	chordal completion
complétion d'intervalles	interval completion

décomposition arborescente	tree decomposition
décomposition de chemins	path decomposition
séparation et évaluation	branch and bound
un triplet astéroïde	an asteroidal triple
sans triplet astéroïde	AT-free
ensemble partiellement ordonné	poset
graphe de couverture	cover graph

Bibliographie

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987. http://dx.doi.org/10.1137/0608024. 29

[AP90] Gur Saran Adhar and Shietung Peng. Parallel algorithms for cographs and parity graphs with applications. *Journal of Algorithms*, 11:252–284, 1990. 14

[BCdMR05] Anne Bergeron, Cédric Chauve, Fabien de Montgolfier, and Mathieu Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. In *Proceedings of the thirteenth Annual European Symposium on Algorithms (ESA'05)*, 2005. http://www-igm.univ-mlv.fr/~raffinot/ftp/common-interval-12-April-05.pdf.gz. 14

[BCHP03] Anna Bretscher, Derek G. Corneil, Michel Habib, and Christophe Paul. A simple linear time LexBFS cograph recognition algorithm. In *Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'03)*, volume 2880 of *Lecture Notes in Computer Science*, pages 119–130, 2003. http://books.google.fr/books?hl=fr&lr=&id=PuHeuUucopEC&oi=fnd&pg=PA119&ots=Y8HsPhrY4M&sig=HjiXf0LLDMbqElF6ByE60JVryiw, http://dx.doi.org/10.1007/b93953. 14, 17

[BCHP08] Anna Bretscher, Derek G. Corneil, Michel Habib, and Christophe Paul. A simple linear time LexBFS cograph recognition algorithm. *SIAM Journal on Discrete Mathematics*, 22(4):1277–1296, 2008. http://www.liafa.jussieu.fr/~habib/Documents/cograph.ps. 14, 17

[Ber94] Claude Berge. Qui a tué le duc de Densmore ? *Bibliothèque Oulipienne*, 67, 1994. réédition Castor Astral, 2000. 25

[BL75] Kellogg S. Booth and George S. Lueker. Linear algorithms to recognize interval graphs and test for the consecutive ones property. In *Proceedings of the seventh Annual ACM Symposium on Theory of Computing (STOC'75)*, pages 255–265, 1975. http://dx.doi.org/10.1145/800116.803776. 23

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Science*, 13:335–379, 1976. 23

[Bla78] Andreas Blass. Graphs with unique maximal clumpings. *Journal of Graph Theory*, 2(1):19–24, 1978. 14

[BM83] Hermann Buer and Rolf H. Möhring. A fast algorithm for the decomposition of graphs and posets. *Mathematics of Operations Research*, 8:170–184, 1983. 14

[Bod05] Hans L. Bodlaender. Discovering treewidth. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, volume 3381 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2005. http://citeseer.ist.psu.edu/bodlaender05discovering.html. 26, 29

[Bri93] Graham Brightwell. On the complexity of diagram testing. *Order*, 10(4):297–303, 1993. http://dx.doi.org/10.1007/BF01108825. 35

[BT06] Graham Brightwell and William T. Trotter. Great book of posets, 2006. book in preparation, http://www.math.gatech.edu/~trotter/math-8823-Fa06/8823-Fa06-TopPage.htm. 33

[BTTvL97] Hans L. Bodlaender, Richard Tan, Dimitrios M. Thilikos, and Jan van Leeuwen. On interval routing schemes and treewidth. *Information and Computation*, 139:92–109, 1997. http://citeseer.ist.psu.edu/188284.html. 31

[BV95] Paola Bonizzoni and Gianluca Della Vedova. Modular decomposition of hypergraphs. In *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science (WG'95)*, volume 1017 of *Lecture Notes in Computer Science*, pages 303–317. Springer-Verlag, 1995. http://dx.doi.org/10.1007/3-540-60618-1_84. 14

[BV99] Paola Bonizzoni and Gianluca Della Vedova. An algorithm to compute the modular decomposition of hypergraphs. *Journal of Algorithms*, 32(2):65–86, 1999. http://citeseer.ist.psu.edu/506177.html. 14

[BW91] Graham Brightwell and Peter Winkler. Counting linear extensions is #P-complete. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC'91)*, pages 175–181, 1991. http://www.math.dartmouth.edu/~pw/papers/sharpstoc.ps. 35

[CC82] Alain Cardon and Maxime Crochemore. Partitioning a graph in $o(|a|log_2|v|)$. *Theoretical Computer Science*, 19(1):85–98, 1982. http://dx.doi.org/10.1016/0304-3975(82)90016-0. 15

[CDHP01] Derek G. Corneil, Feodor F. Dragan, Michel Habib, and Christophe Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2-3):143–166, 2001. http://www.lirmm.fr/~paul/Biblio/Postscript/Diam-new.ps. 20

[CDK03] Derek G. Corneil, Feodor F. Dragan, and Ekkehard Köhler. On the power of BFS to determine a graph's diameter. *Networks*, 42(4):209–222, 2003. http://www.cs.kent.edu/~dragan/LBFS-Power.pdf. 20, 21

[CEdFK98] Márcia R. Cerioli, Hazel Everett, Celina M.H. de Figueiredo, and Sulamita Klein. The homogeneous set sandwich problem. *Information Processing Letters*, 67(1):31–35, 1998. http://citeseer.ist.psu.edu/190684.html. 15

[CG70] Derek G. Corneil and Calvin C. Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the ACM (JACM)*, 17(1):51–64, 1970. http://dx.doi.org/10.1145/321556.321562. 15

[CH93] Alain Cournier and Michel Habib. An efficient algorithm to recognize prime undirected graphs. In *Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'92)*, volume 657 of *Lecture Notes in Computer Science*, pages 212–224. Springer-Verlag, 1993. http://dx.doi.org/10.1007/3-540-56402-0_49. 14

[CH94] Alain Cournier and Michel Habib. A new linear algorithm for modular decomposition. In *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming (CAAP'94)*, volume 787 of *Lecture Notes in Computer Science*, pages 68–84. Springer-Verlag, 1994. http://dx.doi.org/10.1007/BFb0017474. 14

[CH97] Christian Capelle and Michel Habib. Graph decompositions and factorizing permutations. In *Proceedings of the fifth Israeli Symposium on the Theory of Computing and Systems (ISTCS'97)*, 1997. http://citeseer.ist.psu.edu/43689.html. 14, 17

[CHdM02] Christian Capelle, Michel Habib, and Fabien de Montgolfier. Graph decomposition and factorizing permutations. *Discrete Mathematics and Theoretical Computer Science*, 5(1), 2002. http://www.liafa.jussieu.fr/~fm/publications/dmtcs.ps. 14, 17

[CHM81] Michel Chein, Michel Habib, and Marie-Catherine Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37:35–50, 1981. 9, 10

[CI98] Alain Cournier and Pierre Ille. Minimal indecomposable graphs. *Discrete Mathematics*, 183(1-3):61–80, 1998. http://dx.doi.org/10.1016/S0012-365X(97)00077-0. 11

[CMR98] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique width. In *Proceedings of the 24th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'98)*, volume 1517 of *Lecture Notes in Computer Science*, pages 1–16, 1998. http://citeseer.ist.psu.edu/courcelle99linear.html. 31

[Cor04] Derek G. Corneil. Lexicographic breadth first search - a survey. In *Proceedings of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *Lecture Notes in Computer Science*, pages 1–19, 2004. http://books.google.fr/books?hl=fr&lr=&id=ZPDqRTRxITsC&oi=fnd&pg=PA1&ots=OMtnEIG_1y&sig=wuvqtuM50-5bKfYATkZNxHr49gU, http://dx.doi.org/10.1007/b104584. 18

[COS98] Derek G. Corneil, Stephan Olariu, and Lorna K. Stewart. The ultimate interval graph recognition algorithm? In *Proceedings of the ninth annual ACM-SIAM Symposium on Discrete algorithms (SODA'98)*, pages 175–180, 1998. http://citeseer.ist.psu.edu/314904.html. 23

[CPS85] Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985. http://dx.doi.org/10.1137/0214065. 12, 14

[Cun82] William H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic and Discrete Methods*, 3(2):214–228, 1982. http://dx.doi.org/10.1137/0603021. 14

[DGM97] Elias Dahlhaus, Jens Gustedt, and Ross M. McConnell. Efficient and practical modular decomposition. In *Proceedings of the eigth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 26–35, 1997. http://www.cs.colostate.edu/~rmm/DGM97.pdf. 14

[DGM01] Elias Dahlhaus, Jens Gustedt, and Ross M. McConnell. Efficient and practical algorithms for sequential modular decomposition. *Journal of Algorithms*, 41(2):360–387, 2001. http://www.cs.colostate.edu/~rmm/linearDecompJournal.ps. 14, 17

[DGM02] Elias Dahlhaus, Jens Gustedt, and Ross M. McConnell. Partially complemented representations of digraphs. *Discrete Mathematics and Theoretical Computer Science*, 5(1):147–168, 2002. http://www.emis.ams.org/journals/DMTCS/volumes/abstracts/pdfpapers/dm050110.pdf. 7, 14

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976. 15

[Dir61] Gabriel A. Dirac. On rigid circuit graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 25, Universität Hamburg, 1961. 21, 22, 27

[DKV02] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proceedings of the Eight International Conference on Principles and Practice of Constraint Programming (CP'02)*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer-Verlag, 2002. http://citeseer.ist.psu.edu/525259.html. 31

[DM41] Ben Dushnik and Edwin W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63:600–610, 1941. http://www.jstor.org/view/00029327/di994279/99p0366k/0. 33, 35, 36

[Dra05] Feodor F. Dragan. Estimating all pairs shortest paths in restricted graph families: a unified approach. *Journal of Algorithms*, 57:1–21, 2005. http://www.cs.kent.edu/~dragan/APASP-SpGr.pdf. 21

[EGMS94] Andrzej Ehrenfeucht, Harold N. Gabow, Ross M. McConnell, and Stephen J. Sullivan. An $O(n^2)$ divide-and-conquer algorithm for the prime tree decomposition of two-structures and modular decomposition of graphs. *Journal of Algorithms*, 16(2):283–294, 1994. 14

[EHR94] Andrzej Ehrenfeucht, Tero Harju, and Grzegorz Rozenberg. Incremental construction of 2-structures. *Discrete Mathematics*, 128(1):113–141, 1994. 14, 17

[FG69] Delbert R. Fulkerson and O. A. Gross. Incidence matrices, interval graphs, and seriation in archaeology. *Pacific Journal of Mathematics*, 28:565–570, 1969. http://projecteuclid.org/Dienst/UI/1.0/Summarize/euclid.pjm/1102995572. 23, 27

[FG04] Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1-3):3–31, 2004. http://www2.informatik.hu-berlin.de/~grohe/pub/mc.ps. 31

[Fis85] Peter C. Fishburn. *Interval Orders and Interval Graphs*. John Wiley & Sons, 1985. 33

[Fre90] Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI'90)*, pages 4–9, 1990. 31

[Gal67] Tibor Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18(1-2):25–66, 1967. http://dx.doi.org/10.1007/BF02020961. 9, 36

[Gav74] Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory Series B*, 16:47–56, 1974. http://dx.doi.org/10.1016/0095-8956(74)90094-X. 27

[GH64] Paul C. Gilmore and Alan J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964. 26

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979. 29, 35

[GJS76] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. 35

[GKS95] Martin C. Golumbic, Haim Kaplan, and Ron Shamir. Graph sandwich problems. *Journal of Algorithms*, 19(3):449–473, 1995. http://citeseer.ist.psu.edu/golumbic94graph.html. 14, 15

[GLS01] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: a survey. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS'01)*, volume 2136 of *Lecture Notes in Computer Science*, pages 37–57. Springer-Verlag, 2001. http://www.springerlink.com/index/Y8R9EL5KQ21MU73M.pdf. 31

[Gol80] Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980. 18, 36

[Hab81] Michel Habib. *Substitution des structures combinatoires, théorie et algorithmes*. PhD thesis, Université Pierre et Marie Curie, 1981. http://catalogue.bnf.fr/ark:/12148/cb34772090v/PUBLIC. 14

[HdMP04] Michel Habib, Fabien de Montgolfier, and Christophe Paul. A simple linear-time modular decomposition algorithm for graphs, using order extension. In *Ninth Scandinavian Workshop on Algorithm Theory (SWAT'04)*, 2004. http://www.liafa.jussieu.fr/~fm/publications/swat04.pdf. 14

[HHS95] Michel Habib, Marianne Huchard, and Jeremy P. Spinrad. A linear algorithm to decompose inheritance graphs into modules. *Algorithmica*, 13:573–591, 1995. http://www.springerlink.com/index/V3384M611187NL12.pdf. 14

[HLP03] Michel Habib, Emmanuelle Lebhar, and Christophe Paul. A note on finding all homogeneous set sandwiches. *Information Processing Letters*, 87(3):147–151, 2003. http://hal-lirmm.ccsd.cnrs.fr/lirmm-00090365. 15

[HM79] Michel Habib and Marie-Catherine Maurer. On the X-Join decomposition for undirected graphs. *Discrete Applied Mathematics*, 3:201–207, 1979. 14, 36

[HM91] Wen-Lian Hsu and Tze-Heng Ma. Substitution decomposition on chordal graphs and applications. In *Proceedings of the second International Symposium on Algorithms (ISA'91)*, volume 557 of *Lecture Notes in Computer Science*, pages 52–60. Springer-Verlag, 1991. http://dx.doi.org/10.1007/3-540-54945-5_49. 23

[HM99] Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004–1020, 1999. http://dx.doi.org/10.1137/S0097539792224814. 23

[Hoa61] Charles A. R. Hoare. Algorithm 63: Partition, algorithm 64: Quicksort, algorithm 65: Find. *Communications of the ACM*, 4(7):321–322, 1961. http://dx.doi.org/10.1145/366622.366642. 15

[Hop71] John E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi and A. Paz, editors, *Theory of machines and computations, Proceedings of an International Symposium on the Theory of Machines and Computations*, pages 189–196. Academic Press, 1971. 15

[HP01] Michel Habib and Christophe Paul. A simple linear time algorithm for cograph recognition. http://citeseer.ist.psu.edu/habib01simple.html, 2001. 14, 17

[HP05] Michel Habib and Christophe Paul. A simple linear time algorithm for cograph recognition. *Discrete Applied Mathematics*, 145(2):183–197, 2005. http://dx.doi.org/10.1016/j.dam.2004.01.011. 14

[HPMV00] Michel Habib, Christophe Paul, Ross M. McConnell, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234:59–84, 2000. http://citeseer.ist.psu.edu/179874.html. 23, 24

[HPV99] Michel Habib, Christophe Paul, and Laurent Viennot. Partition refinement techniques: an interesting algorithmic toolkit. *International Journal of Foundations of Computer Science*, 10(2):147–170, 1999. http://gyroweb.inria.fr/~viennot/postscripts/ijfcs00.ps.gz. 14, 17

[Hsu93] Wen-Lian Hsu. A simple test for interval graphs. In *Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'92)*, volume 657 of *Lecture Notes in Computer Science*, pages 11–16. Springer-Verlag, 1993. http://dx.doi.org/10.1007/3-540-56402-0_31. 23

[Jor69] Camille Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, pages 185–190, 1869. http://www.digizeitschriften.de/index.php?id=loader&tx_jkDigiTools_pi1[IDDOC]=510097. 19

[JSC72] Lee O. James, Ralph G. Stanton, and Donald D. Cowan. Graph decomposition for undirected graphs. In F. Hoffman, editor, *Proceedings of the third Southeastern International Conference on Combinatorics, Graph Theory, and Computing (CGTC'72)*, pages 281–290. R. B. Levow eds., 1972. 14

[KF79a] T. Kashiwabara and T. Fujisawa. An NP-complete problem on interval graphs. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'79)*, pages 82–83, 1979. 29

[KF79b] T. Kashiwabara and T. Fujisawa. NP-completeness of the problem of finding a minimum-clique number interval graph containing a given graph as a subgraph. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'79)*, pages 82–83, 1979. 29

[Klo93] Ton Kloks. *Treewidth*. PhD thesis, Utrecht University, Utrecht, The Netherlands, 1993. 29

[KM86] Norbert Korte and Rolf H. Möhring. A simple linear-time algorithm to recognize interval graphs. In *Proceedings of the twelfth International Workshop on Graph-Theoretic Concepts in Computer Science (WG'86)*, volume 246 of *Lecture Notes in Computer Science*, pages 1–16, 1986. dx.doi.org/10.1007/3-540-17218-1_45. 23

[KM89] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989. http://dx.doi.org/10.1137/0218005. 23

[KT06] Jon Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006. 3

[LB62] Cornelis G. Lekkerkerker and J. Ch. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51:45–64, 1962. http://matwbn.icm.edu.pl/ksiazki/fm/fm51/fm5115.pdf. 23

[LO91] Rong Lin and Stephan Olariu. An NC recognition algorithm for cographs. *Journal of Parallel and Distributed Computing*, 13(1):76–90, 1991. 14

[Lov05] László Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43:75–86, 2005. http://www.ams.org/bull/2006-43-01/S0273-0979-05-01088-8/. 31

[Mau77] Marie-Catherine Maurer. *Joints et décompositions premières dans les graphes*. PhD thesis, Université Pierre et Marie Curie (Paris VI), 1977. 14

[McC87] Carolyn McCreary. *An algorithm for parsing a graph grammar*. PhD thesis, University of Colorado, 1987. 14

[McC95] Ross M. McConnell. An $O(n^2)$ incremental algorithm for modular decomposition of graphs and 2-structures. *Algorithmica*, 14:209–227, 1995. http://www.cs.colostate.edu/~rmm/incralg.pdf. 14

[MdM05] Ross M. McConnell and Fabien de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Mathematics*, 2(145):189–209, 2005. http://www.cs.colostate.edu/~rmm/digraphDecomp.pdf. 14

[MP01] Frederic Maffray and Myriam Preissmann. A translation of Tibor Gallai's paper: Transitiv orientierbare Graphen. In J.L. Ramirez-Alfonsin and B.A. Reed, editors, *Perfect graphs*, pages 25–66. J. Wiley, 2001. 9

[MR84] Rolf H. Möhring and Franz J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984. 9

[MS89] John H. Muller and Jeremy Spinrad. Incremental modular decomposition. *Journal of the ACM (JACM)*, 36(1):1–19, 1989. http://dx.doi.org/10.1145/58562.59300. 14

[MS94] Ross M. McConnell and Jeremy P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94)*, pages 536–545, 1994. http://www.cs.colostate.edu/~rmm/linModDecomp.pdf. 14

[MS97] Ross M. McConnell and Jeremy P. Spinrad. Linear-time transitive orientation. In *Proceedings of the eigth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, 1997. http://www.cs.colostate.edu/~rmm/linTOSODA.pdf. 14

[MS99] Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201:189–241, 1999. http://www.cs.colostate.edu/~rmm/linto.pdf. 14

[MS00] Ross M. McConnell and Jeremy P. Spinrad. Ordered vertex partitioning. *Discrete Mathematics and Theoretical Computer Science*, 4:45–60, 2000. http://www.cs.colostate.edu/~rmm/dm040104.pdf. 14, 17

[MV96] Michel Morvan and Laurent Viennot. Parallel comparability graph recognition and modular decomposition. In *Proceedings of the thirteenth Annual Symposium on Theoretical Aspects of Computer Science (STACS'96)*, volume 1046 of *Lecture Notes in Computer Science*, pages 169–180. Springer-Verlag, 1996. http://www.liafa.jussieu.fr/web9/rapportrech/rapportsWeb/1995/stacs96.pdf. 14

[NR87] Jaroslav Nešetřil and Vojtěch Rödl. Complexity of diagrams. *Order*, 3:321–330, 1987. 34, 35

[NR93] Jaroslav Nešetřil and Vojtěch Rödl. Complexity of diagrams, errata. *Order*, 10:393, 1993. 35

[NR95] Jaroslav Nešetřil and Vojtěch Rödl. More on the complexity of cover graphs. *Commentationes mathematicae Universitatis Carolinae*, 36(2):271–280, 1995. http://www.karlin.mff.cuni.cz/cmuc/pdf/cmuc9502/nesetril.pdf. 35

[Ola91] Stephan Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37(1):21–25, 1991. http://dx.doi.org/10.1016/0020-0190(91)90245-D. 24

[Pau06] Christophe Paul. Aspects algorithmiques de la décomposition modulaire. Université Montpellier II, 2006. Habilitation thesis. 13, 17

[PT87] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987. http://dx.doi.org/10.1137/0216062. 15, 16

[Ros74] Donald J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, 7:317–322, 1974. 28

[RR90] Ganesan Ramalingam and Chandrasekaran Pandu Rangan. New sequential and parallel algorithms for interval graph recognition. *Information Processing Letters*, 34(4):215–219, 1990. http://dx.doi.org/10.1016/0020-0190(90)90163-R. 23

[RS83] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory Series B*, 35:39–61, 1983. 29

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986. 29

[RTL76] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. http://dx.doi.org/10.1137/0205021. 23, 26

[Sim92] Klaus Simon. A new simple linear algorithm to recognize interval graphs. In *Proceedings of the International Workshop on Computational Geometry (CG'91)*, volume 553 of *Lecture Notes in Computer Science*, pages 289–308. Springer-Verlag, 1992. http://dx.doi.org/10.1007/3-540-54891-2_22. 23

[SMTW01] Fu-Long Yeh Shyue-Ming Tang and Yue-Li Wang. An efficient algorithm for solving the homogeneous set sandwich problem. *Information Processing Letters*, 77(1):17–22, 2001. http://dx.doi.org/10.1016/S0020-0190(00)00145-9. 15

[Spi82] Jeremy P. Spinrad. *Two-dimensional partial orders*. PhD thesis, Princeton University, 1982. 14

[SS04] Ron Shamir and Roded Sharan. A fully-dynamic algorithm for modular decomposition and recognition of cographs. *Discrete Applied Mathematics*, 136(2-3):329–340, 2004. http://www.math.tau.ac.il/~rshamir/papers/dcog_DAM.ps. 14

[ST93a] James H. Schmerl and William T. Trotter. Critically indecomposable partially ordered sets, graphs, tournaments and other binary relational structures. *Discrete Mathematics*, 113:191–205, 1993. 12

[ST93b] Paul D. Seymour and Robin Thomas. Graph searching, and a min-max theorem for tree-width. *Journal of Combinatorial Theory Series B*, 58:22–33, 1993. 31

[Ste82] George Steiner. *Machine scheduling with precedence constraints*. PhD thesis, Department of Combinatorics and Optimization, University of Waterloo, 1982. 14

[Szp30] Edward Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae*, 16:386–389, 1930. http://matwbn.icm.edu.pl/ksiazki/fm/fm16/fm16125.pdf. 35

[TCHP08] Marc Tedder, Derek Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP'08)*, volume 5125 of *Lecture Notes in Computer Science*, pages 634–645, 2008. http://arxiv.org/abs/0710.3901. 14

[Tho98] Mikkel Thorup. Structured programs have small tree-width and good register allocation. *Information and Computation*, 142:159–181, 1998. http://www.diku.dk/hjemmesider/ansatte/mthorup/PAPERS/register.ps.gz. 30

[UY00] Takeaki Uno and Mutsunori Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 14:209–227, 2000. http://citeseer.ist.psu.edu/uno00fast.html. 13, 14

[Yan81] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79, 1981. http://dx.doi.org/10.1137/0602010. 29, 35