Université Paris Diderot

Ufr d'Informatique

Sciences Mathématiques de Paris Centre

Doctorat

Sur quelques généralisations polynomiales de la décomposition modulaire

On some polynomial generalizations of modular decomposition

Vincent Limouzy

Thèse dirigée par Michel Habib

Soutenue le 3 décembre 2008

Jury

Pierre Fraignaud : Directeur de Recherche, Cnrs – Liafa . . . . . Examinateur
Martin C. Golumbic : Professeur, University of Haïfa . . . . . Examinateur
Michel Habib : Professeur, Université Paris Diredot – Liafa . . . . . Directeur
Pinar Heggernes : Professeur, University of Bergen . . . . . Examinateur
Frédéric Maffray : Directeur de Recherche, Cnrs – G-Scop . . . . . Examinateur
Matthieu Latapy : Chargé de Recherche, Cnrs – Lip 6 . . . . . Rapporteur
Ioan Todinca : Professeur, Université d'Orléans – Lifo . . . . . Rapporteur

Remerciements

« This is the end
Beautiful friend
This is the end
My only friend, the end »
The Doors – The end

Pour qui me connaît sait que par nature je ne suis pas quelqu'un de très expressif. Je profite de ces quelques lignes pour exprimer ma reconnaissance à toutes celles et ceux qui ont de près ou de loin contribué à ce que cette thèse aboutisse.

Je tiens à remercier Matthieu Latapy et Ioan Todinca pour avoir accepter la lourde tâche de relire le présent manuscrit. Je tiens également à les remercier à titre individuel. Chacun à leur manière m'ont fait entrevoir des points de vues différents de la discipline.

Je tiens à remercier Pinar Heggernes, Frédéric Maffray et Pierre Fraignaud pour avoir accepté de prendre part à mon jury. Leurs nombreuses questions vont me tenir occupé quelques années.

Je suis également reconnaissant envers Martin Golumbic à plus d'un titre. Tout d'abord, pour avoir accepté, une fois la surprise passée devant l'incongruité de ma requête, de me rencontrer à Lausanne lors de cet incroyable symposium. Ensuite pour avoir accepté de participer à mon jury ainsi que pour l'intérêt qu'il a porté à l'égard de mes travaux. Enfin pour m'avoir offert l'opportunité de travailler ensemble dans un avenir proche.

On peut dire que cette thèse s'est terminée à distance, en effet grâce au chaleureux accueil de Derek Corneil. J'ai pu prendre le recul nécessaire sur les travaux menés en thèse. Ce séjour à Toronto m'a permis d'entrevoir d'autres façons de travailler, d'envisager les choses différemment. Je tiens à remercier Derek pour son accueil, ses précieux conseils et ses connaissances encyclopédiques sur les graphes et leurs algorithmes.

Je remercie ceux qui ont fait en sorte que mon atterrissage au LIAFA se fasse en douceur, je pense en particulier à Fab, Borg, Pascal et Mohsen ainsi que les membres du bureau 6A51, ainsi que Christophe, Clémence et les membres de l'équipe et du labo. En particulier Noëlle, son incroyable efficacité et sa gentillesse ont grandement simplifié ma vie au labo.

Cette thèse repose en grande partie sur les résultats obtenus avec mes co-auteurs. Je tiens à remercier Binh-Minh qui lors de mon arrivée au LIRMM en temps que simple stagiaire a su trouver les mots pour me mettre en confiance, mais aussi pour toutes ces discussions, parfois scientifiques, que nous avons eu au cours de ces années... Sa rigueur pour écrire des articles s'avérait parfois pénible s'est révélée être une grande qualité au moment de rédiger le présent manuscrit. Je me souviendrai toujours de cette séances de travail qui s'est tenue à Levico autour de quelques verres de Grappa !

Je tiens également à remercier Fabien pour son enthousiasme débordant sa vivacité. Ça a été un plaisir de travailler et d'enseigner avec lui. Il a su me transmettre certaines ficelles que je n'aurais certainement jamais trouvées par moi même.

Je voudrais également remercier Pierre et Mathieu qui chacun à leur façon m'ont fait profité de leurs connaissances et de leur expériences de domaines qui n'était pas forcément les miens.

Cette thèse ne serait pas complète sans l'influence de Michaël. Je tiens à le remercier pour m'avoir montré une (autre) façon d'appréhender les décompositions et les familles bipartitives.

Je tiens à remercier ma famille qui m'a patiemment supporté, dans tous les sens du terme, ces trois longues années. J'ai bien conscience que sans eux, cela n'aurait pas été possible.

Il me reste encore à remercier Michel, à plusieurs points de vues. Je réalise aujourd'hui la chance qu'il m'a donné de venir faire ma thèse au LIAFA. J'ai ainsi eu l'opportunité de découvrir un monde assez nouveau. Tout au long de ces trois années, il a su m'encadrer sans trop de contraintes tout en montrant les pistes à explorer. Et au delà des aspects scientifiques il y a les aspects humains, et je me sens assez privilégié d'avoir pu travailler mais également régater avec lui !

Enfin, une liste de gens qui ont influencé et contribué à élargir l'étendue de mes connaissances et qui ne rentraient pas forcément dans les catégories précédentes : les deux Stéphanes, les deux Christophes, Nicolas Trotignon, David Ilcinkas, Emmanuelle Lebhar, Nicolas Nisse, Benjamin Lévêque, Alexandre Pinlou, Daniel Gonçalves, Mamadou Kante, Philippe Gambette, Jean Daligault, Mauricio Soto, Xavier Koegler, Hervé Baumann et enfin Mathilde Bouvel.

Enfin, je tiens à remercier Leïla, pour toutes les qualités que je te connaissais, j'ai beaucoup appris à ton contact. Tu m'as aussi appris que certaines questions resteront à jamais sans réponse.

Sommaire

CHAPITRE 1

Introduction

Plan

1. Contexte

Cette thèse se situe à la croisée des chemins de l'algorithmique des graphes et la théorie des graphes.

Dans la littérature nous distinguons plusieurs types de décompositions, dans un premier temps les décompositions paramétrées qui, en un sens, mesurent la «complexité» du graphe, on pourra retenir la largeur arborescente (tree-width) due à Robertson et Seymour (1986a), ou la largeur de branche (branch-width) Robertson et Seymour (1991), ou encore la largeur de clique (clique-width) introduite par Courcelle et al. (1993) ou encore la largeur NLC (pour Node Labelled Controlled) introduite par Wanke (1994). Ces deux dernières décompositions transforment le graphe en un expression grammaticale exprimé sur un arbre. Récemment Oum (2005) a introduit une nouvelle décomposition : la largeur de rang (rank-width) qui englobe, d'une certaine manière, l'ensemble des décompositions présentés ci-dessus. Toutes ces décompositions dépendent d'un paramètre k, et pour tout graphe il est possible de trouver une décomposition où k prendra diverses valeurs. L'intérêt de ces décompositions, est qu'elles permettent de capturer la structure du graphe considéré, et par conséquent en fixant le paramètre k qui mesure cette complexité, il est possible de résoudre des problèmes, en général, difficiles en temps polynomial, à la condition que ce paramètre soit fixé, on pourra se rapporter à Arnborg et Proskurowski (1989) ; Arnborg et al. (1991) pour plus de détails.

Dans un second temps, nous considérons les décompositions dites «exactes» dans la mesure où la décomposition consiste à trouver un «motif» bien particulier, et décomposer seulement lorsque l'on rencontre ce «motif». La décomposition modulaire, la décomposition en coupes, et leurs généralisations respectives (2-module, 2-join, k-modules, Bimodules, ...) entrent dans ce cadre. Le principal désavantage par rapport à la famille de décomposition précédente, est que pour celles-ci le nombres de graphes qui n'admettent pas ce motif est très grand. Il nous est donc impossible de décomposer ces graphes. Nous appelons ces graphes, des graphes premier vis à vis d'une décomposition.

Toutefois, le tableau n'est pas si noir, en effet même si les contraintes exigées par ces décompositions semblent fortes elles se révèlent être des outil très puissants sur les graphes qu'elles permettent de décomposer et ainsi d'apprendre beaucoup sur la structure du graphe, ou de la famille de graphes considérée.

2. Pourquoi décomposer ?

Les décompositions de graphes ont de nombreuses applications, et ce, à tout point de vue, en effet les décompositions servent aussi bien en biologie (ou bioinformatique) pour modéliser et résoudre les problèmes rencontrés, nous pourrons d'ailleurs consulter les travaux de Gagneur et al. (2004) sur les interactions entre protéines. La décomposition modulaire est également utilisée en phylogénie. Mais elles sont également utilisées en sociologie, essentiellement pour modéliser les rôles des différent acteurs dans une population et ainsi permettre un certaine classification et détecter les communautés qui se dégagent. Nous pourrons consulter pour cela les travaux de White et Reitz (1983), ou encore les travaux de Everett et Borgatti (1991) où le concept de rôle est étudié et formalisé. Par la suite, Fiala et Paulusma (2003, 2005) ont montré que le calcul de ce genre de structure est difficile, *i.e.* **NP**-Complet.

Les décompositions sont également utiles pour la discipline elle même. En effet, les décompositions constituent un formidable outils d'analyse de la structure des graphes[1] et elles s'avèrent utiles lorsque l'on souhaite résoudre des problèmes difficiles et faire face à l'explosion combinatoire. La décomposition permet, en général, de résoudre le problème sur des instances de taille moindre, où le problème peut se révéler être plus facile à traiter, il suffit ensuite de remettre les solutions ensemble. Dans cet esprit Bodlaender et Rotics (2003) ont montré comment la décomposition modulaire pouvait être utilisée pour faciliter le calcul de la largeur arborescente d'un arbre ainsi que le calcul du *Minimum Fill-in*. Nous pourrons également consulter les travaux de Protti et al. (2006) où une fois encore la décomposition modulaire est utilisée pour résoudre certains problèmes de calcul de *clusters*.

[1] L'objet du projet ANR GRAAL.

Les décompositions de graphes sont des outils très puissants, un bon exemple est que les décompositions de graphes ont permis la résolution de deux problèmes majeurs de la théorie des graphes pour les 50 dernières années.

En effet le premier problème est la conjecture de Wagner qui dit que l'ensemble des graphes (non-orientés finis) sont WQO[2] pour la relation des mineurs[3].

Le second problème est la conjecture de Berge qui dit que tout graphe parfait[4] est sans trou[5] ni antitrou impair.

Le premier problème a été résolu par Robertson et Seymour dans leur fameuse série de papiers sur les mineurs de graphes. Leur solution utilise de nombreux résultats intermédiaires qui tous pour la plupart font appel à deux décompositions qu'ils ont introduites. La première étant la largeur arborescente (tree-width, *cf.* Robertson et Seymour (1986a)) la seconde étant la décomposition en branches (branch-width). Entre autres, comme résultat intermédiaire, ils ont montré que les graphes planaires étaient WQO (*cf.* Robertson et Seymour (1986b)) et les graphes de largeur arborescente bornée étaient également WQO (*cf.* Robertson et Seymour (1990)). Le résultat final est apparu en 2004 après 20 ans de traque : Robertson et Seymour (2004).

Le second problème a été résolu par Chudnovsky, Robertson, Seymour et Thomas en 2003, et une fois de plus la solution fait appel à des décompositions de graphes. Pour ce faire ils montrent qu'un graphe est parfait si et seulement si il répond au schéma suivant : Soit le graphe est un graphe basique, un classe de graphe bien particulière. Dans le cas des graphes parfais il s'agit des graphes bipartis, des line graphes d'un graphe biparti ou leur complémentaire, ou encore un double split graphe. Si le graphe n'est pas basique et parfait alors il admet :

– G ou son complémentaire admet un 2-join, soit
– G ou son complémentaire admet une partition *skew*, soit
– G ou son complémentaire admet un 2-module (Paire homogène).

Il s'avère finalement que les 2-modules ne sont pas nécessaire à leur preuve... Le résultat peut être trouvé dans Chudnovsky et al. (2006).

3. Les décompositions "exactes" : un Aperçu

Afin de comprendre les généralisations de la décomposition modulaire qui seront présentées dans les chapitres suivants, nous présentons dans cette section quelques décompositions de graphes, dites "*exactes*", qui généralisent d'une certaine manière la décomposition modulaire. Nous verrons ainsi quels sont les points communs à toutes ces décompositions.

[2]Well Quasi Order.
[3]contraction d'arêtes, suppression d'arêtes et de sommets isolés.
[4]Un graphe est parfait si pour tous ses sous graphes nous avons $\chi = \omega$.
[5]Un trou est un cycle sans corde.

Définition 1.1 (Module Gallai (1967)). Un module dans un graphe est un sous-ensemble M de V, tel que tout sommet à l'extérieur de M est :

– soit connecté à tous les sommets de M,

– soit déconnecté de tous les sommets de M.

Un exemple est donné en figure 1.10.

Définition 1.2 (2-Module Chvátal et Sbihi (1987) ; Everett et al. (1997)). Un 2-module dans un graphe est un sous-ensemble M de V qui peut être bipartitionné en $\{M_1, M_2\}$ tel que pour tout sommet $x \in V \setminus M$ on ait :

$$N_M(x) = \begin{cases} M_1 \\ M_2 \\ M \\ \varnothing \end{cases}$$

Un exemple est donné en figure 1.1.

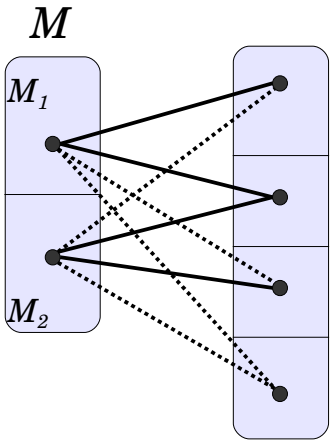


Figure 1.1. Vision schématique d'un 2-Module M.

Dans le but de paramétrer la décomposition modulaire, Rao (2006) a introduit les k-modules, pour ensuite introduire la largeur modulaire.

Définition 1.3 (k-Module, Rao (2006)). Soit k, un entier positif, et soit $G = (V, E)$ un graphe non orienté. Un k-module de G est un sous-ensemble M de V qui peut être partionné en k parties $M_1, \ldots, M_k$ telle que pour tout $u, v \in M_i$ nous avons $N(u) \setminus M = N(v) \setminus M$. pour tout $i \in \{1, \ldots, k\}$.

Remarquons que les modules, les 2-modules et les k-modules sont complémentés, autrement dit si M est un module (*resp.* 2-module, k-module) de G, M est également un module (*resp.* 2-module, k-module) de $\bar{G}$.

Les bimodules ont été introduits par Montgolfier (2003) afin de généraliser les modules aux graphes bipartis. En effet lorsque l'on considère les graphes bipartis, la décomposition modulaire atteint rapidement ses limites, la structure de cette famille qui interdit des cycle impairs, et par conséquent les graphes bipartis contiennent un grand nombre de P_4.

DÉFINITION 1.4 (Bi-module Montgolfier (2003)). Soit $B = (V_1, V_2, E)$ un graphe biparti. Un bi-module de B est un ensemble $V_1' \cup V_2'$ tel que $V_1' \subseteq V_1$ et $V_2' \subseteq V_2$, tout sommet de $V_1 \setminus V_1'$ est soit complètement connecté à V_2' soit complètement déconnecté. De manière symétrique, tout sommet de $V_2 \setminus V_2'$ est soit complètement connecté à V_1', soit complètement déconnecté.

DÉFINITION 1.5 (Bi-join Montgolfier et Rao (2005a)). Soit $G = (V, E)$ un graphe simple non orienté. Un bi-join dans G,

est une bipartition de V en deux ensembles V_1 et V_2 telle que on puisse bipartitionner V_1 et V_2 en U_1, W_1 et en U_2, W_2. tel que U_1 est complet à U_2 et W_1 est complet à W_2. Ce sont les seules arêtes autorisées entre V_1 et V_2. De plus nous autorisons qu'une des parties U_i ou W_i soit vide.

DÉFINITION 1.6 (Coupe (Split ou 1-join) Cunnigham et Edmonds (1980) ; Cunningham (1982)). Dans un graphe $G = (V, E)$, une coupe consiste en une bipartition de V en $\{V_1, V_2\}$ avec $U_1 \subseteq V_1$ et $U_2 \subseteq V_2$, où tous les sommets de U_1 sont reliés à tous les sommets de U_2. Les arêtes de types $u_{1i}u_{2j}$ sont les seules arêtes autorisées entre V_1 et V_2.

Une coupe est non triviale si $|V_1| \geqslant 2$ et $|V_2| \geqslant 2$. Une illustration de coupe est donnée en figure 1.2.

De nombreuses propriétés sont également exposées, dans la thèse de Lanlignel (2001) et celle de Rao (2006). Cette décomposition a de nombreuses applications. Elle constitue une généralisation naturelle de la décomposition modulaire. Une application intéressante est que, comme dans le cas de graphe de permutation pour la décomposition modulaire, un graphe de cercle [6] admet une unique représentation si et seulement si le graphe est premier pour cette décomposition (*cf.* Bouchet (1985, 1987) ; Gabor et al. (1989)).

Son utilisation permet également de colorer certaines classes de graphe efficacement Rao (2004), ou encore de trouver des cliques maximum dans certaines classes de graphes Cicerone et Di Stefano (1999a,b)[7] efficacement.

DÉFINITION 1.7 (2-join Cornuéjols et Cunningham (1985)). Soit $G = (V, E)$ un graphe, un 2-join propre dans un graphe est une bipartition de V en (V_1, V_2), tel qu'il existe dans

[6] Un graphe de cercle est un graphe qui a pour modèle d'intersection de cordes dans un cercle, les cordes représentent les sommets. Deux sommets sont adjacents si et seulement si les cordes correspondantes se croisent.

[7] corrigé par Rao (2004)

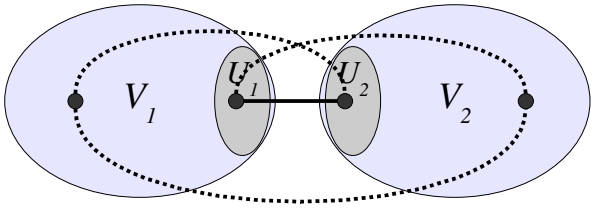


Figure 1.2. Vision schématique d'une coupe dans un graphe.

V_1 deux ensembles disjoints $U_{1,1}$ et $U_{1,2}$, et dans V_2 deux ensembles $U_{2,1}$ et $U_{2,2}$. Où tous les sommets de $U_{1,1}$ sont complètement reliés aux sommets de $U_{2,1}$ et tous les sommets de $U_{1,2}$ sont complètement reliés aux sommets de $U_{2,2}$. Et ce sont les seules arêtes possibles entre V_1 et V_2.

Chaque composante de $G[V_i]$ ($i \in \{1, 2\}$) rencontre à la fois $U_{i,1}$ et $U_{i,2}$.

De plus si $|U_{i,1}| = |U_{i,2}| = 1$, et $G[V_i]$ est un chemin joignant $U_{i,1}$ à $U_{i,2}$ alors il est de longueur impaire ($\geqslant 3$).

Un illustration de 2-join est donnée en figure 1.3.

Cette définition est en fait un cas particulier de la définition donnée par Cornuéjols et Cunningham (1985), cependant, c'est cette décomposition qui est utilisée pour le théorème fort des graphes parfaits. Cette décomposition s'est également avérée utile pour caractériser une classe proche des graphes de Berge (en terme de définition), la classe des graphes sans trou pair Conforti et al. (2001, 2002)

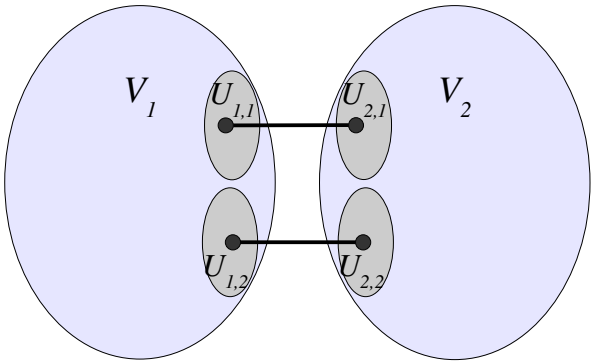


Figure 1.3. Une vision schématique des 2-join

Une fois considérée toutes ces décompositions, où il s'agit le plus souvent de trouver un "*motif*" dans une bipartition, Rusu et Spinrad (2002) se sont intéressés à considérer les décompositions que nous obtenons lorsque l'on interdit un certain motifs biparti. L'approche

est similaire à caractériser une classe de graphes par sous-graphes interdits, à la différence qu'ici, l'objet caractérisé est un structure plus générale à savoir : une décomposition.

Voici la définition formelle :

DÉFINITION 1.8 (Motifs interdits bipartis Rusu et Spinrad (2002)). Soit un graphe $G = (V, E)$, une bipartition ordonnée (V_1, V_2) de V.

Et soit F un graphe biparti de la forme $F = (B_1, B_2, L)$ la bipartition (V_1, V_2) est dite F-free si (V_1, V_2) ne contient pas F comme sous-graphe biparti induit.

PROPOSITION 1.9 (Motif interdits des Modules Rusu et Spinrad (2002)). *Une bipartition (V_1, V_2) correspond à un module (V_1 en l'occurrence) si et seulement si elle ne contient pas de K_2 plus un un sommet isolé (cf. figure 1.4).*

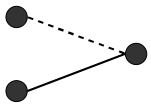
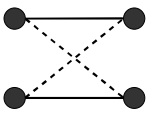
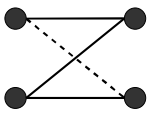


Figure 1.4. Motif interdit pour la décomposition modulaire. Ce motif empêche la partie gauche (*i.e.* les deux sommets à gauche) de former un module.

PROPOSITION 1.10 (Motif interdit pour la décomposition en coupes Rusu et Spinrad (2002)). *Une bipartition (V_1, V_2) correspond à une coupe (split) si et seulement si elle ne contient par $2K_2$ (deux arêtes disjointes) et un P_4. (cf. figure 1.5).*



(a) $2K_2$ deux arêtes disjointes.



(b) P_4 biparti.

Figure 1.5. Motifs interdits pour la décomposition en coupe.

PROPOSITION 1.11 (Motifs interdits pour les 2-modules). *Une bipartition (V_1, V_2) correspond à un 2-module (V_1 en l'occurrence) si et seulement si il ne contient pas de P_5 biparti et de co-P_5 biparti (cf. figure 1.6).*

Cependant pour les 2-joins rien n'est fait, il semble possible de caractériser par motif biparti, au moins deux P_4 biparti disjoints sont interdits, mais ce n'est pas suffisant pour caractériser ce qu'il se passe à l'intérieur d'une partie (la condition sur la longueur des chemins).

(a) P_5 biparti

(b) co-P_5 biparti

Figure 1.6. Motifs bipartis interdits pour la décomposition en 2-Modules. Ces deux motifs empêche la partie gauche (les 3 sommets à gauche) de former un 2-module. Remarquons que ces deux motifs sont complémentaires (vis à vis du bi-complémentaire).

Finalement, un schéma récapitulatif donne les relations d'inclusions des différentes décompositions en figure 1.7.

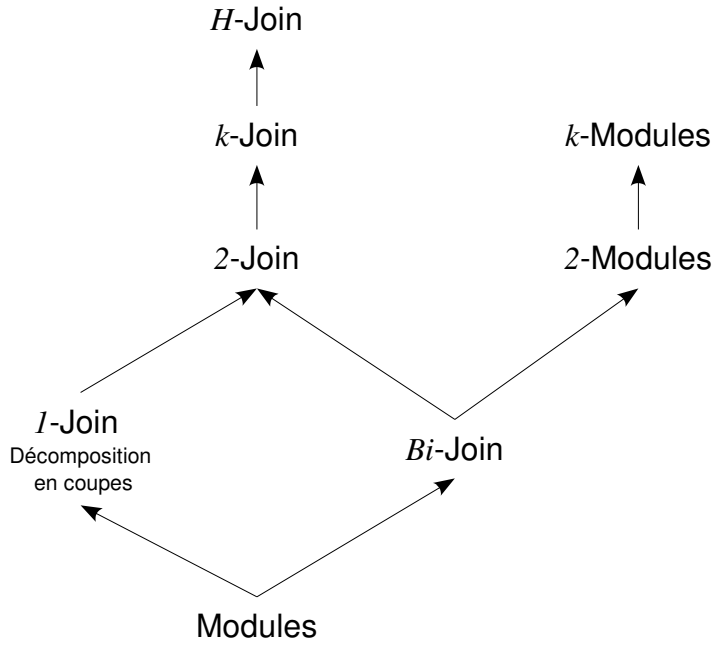


Figure 1.7. Hiérarchie des Décompositions, généralisations de la décomposition modulaire.

Le point commun à toutes ces décompositions, est que le *"comportement"* de l'ensemble de sommet considéré (module, 2-module, ...) est, d'une certaine manière, homogène vis à vis des sommets qui se situent à l'extérieur. Dans les chapitres qui suivent, nous présentons un mécanisme d'abstraction de cette notion d'homogénéité.

4. Rappels sur la Décomposition Modulaire

4.1. Famille des modules et familles de parties.

NOTATION 1.12 (Chevauchement). Soit X un ensemble fini, A et B des sous-ensembles de X. On dit que A et B se chevauchent, noté $A \oslash B$, si :

$$A \cap B \neq \varnothing,\ A \setminus B \neq \varnothing \text{ et } B \setminus A \neq \varnothing.$$

Une illustration est donnée en figure 1.8

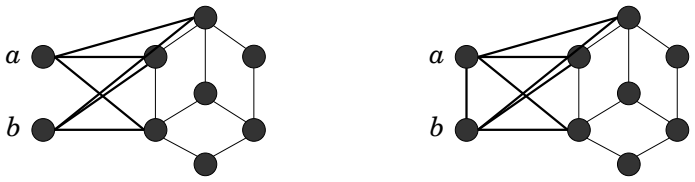
Figure 1.8. A et B se chevauchent : $A \oslash B$

DÉFINITION 1.13 (Jumeaux). Soit $G = (V, E)$ un graphe, et soient x, y deux sommets de G. x et y sont appelés jumeaux si :

$$N_G(x) \setminus \{x\} \cup \{y\} = N_G(y) \setminus \{x\} \cup \{y\}$$

On peut partionner les jumeaux en deux familles (*cf.* figure 1.9),

(1) Vrais jumeaux, si $xy \in E$
(2) Faux jumeaux, si $xy \notin E$



(a) Faux Jumeaux (b) Vrai Jumeaux

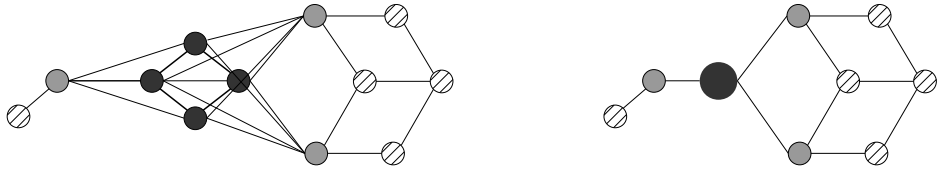
Figure 1.9. a et b sont deux sommets jumeaux.

Nous avons déjà présenté les modules en définition 1.1, afin de faciliter la lecture, nous présentons ici une définition équivalente.

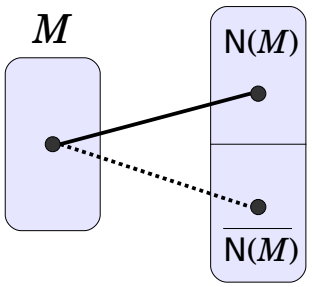
Définition 1.14 (Module). Soit $G = (V, E)$ un graphe fini non orienté. Un module de G est un sous-ensemble de sommets M de V tel que pour tout sommet x extérieur à l'ensemble M (*i.e.* $x \in V \setminus M$) on ait :

- $N_G(x) \cap M = M$ soit
- $N_G(x) \cap M = \varnothing$.

Un illustration du concept de Module est donnée en figure 1.10.



(a) (b)



(c)

Figure 1.10. En (a) un graphe non orienté où les sommets noirs forment un module. En (b) le même graphe où le module (les sommets noirs) ont été contracté en un seul point : le sommet noir. Enfin en (c) une vision schématique des modules, de manière à faire ressortir les relations de voisinages.

Définition 1.15 (Module Fort). Un module M d'un graphe G, est dit fort si aucun autre module M' de G ne le chevauche. En d'autres termes, soit M et M' sont disjoints, soit l'un contient l'autre.

Définition 1.16 (Casseur). Soit $G = (V, E)$ et $X \subseteq V$, un sommet c de V est un casseur de X si il existe $u, v \in X$ tel que $uc \in E$ et $vc \notin E$.

Remarque 1.17. Un ensemble M est un module si et seulement si $V \setminus M$ ne contient aucun casseur.

DÉFINITION 1.18 (Familles partitives Chein et al. (1981)). Soit X un ensemble fini, et $\mathscr{F}$ une famille sur 2^X. La famille $\mathscr{F}$ est partitive si elle satisfait les propriétés suivantes :
Pour toute paire de sous-ensembles A et B tel que $A \Cap B$ on a :

(1) $A \cup B$,

(2) $A \cap B$,

(3) $A \Delta B$,

(4) $A \setminus B$ et

(5) $B \setminus A$ appartiennent à la famille $\mathscr{F}$.

REMARQUE 1.19. Nous pouvons remarquer que les conditions (4) et (5) peuvent être retrouvées à partir de (2) et (3).

La famille des modules d'un graphe non orienté constitue une famille partitive.

DÉFINITION 1.20 (Familles faiblement partitives Chein et al. (1981)). Soit X un ensemble fini, et $\mathscr{F}$ une famille sur 2^X. La famille $\mathscr{F}$ est faiblement partitive si elle satisfait les propriétés suivantes :
Pour toute paire de sous-ensembles A et B tel que $A \Cap B$ nous avons :

(1) $A \cup B$,

(2) $A \cap B$,

(3) $A \setminus B$ et

(4) $B \setminus A$ appartiennent à la famille $\mathscr{F}$.

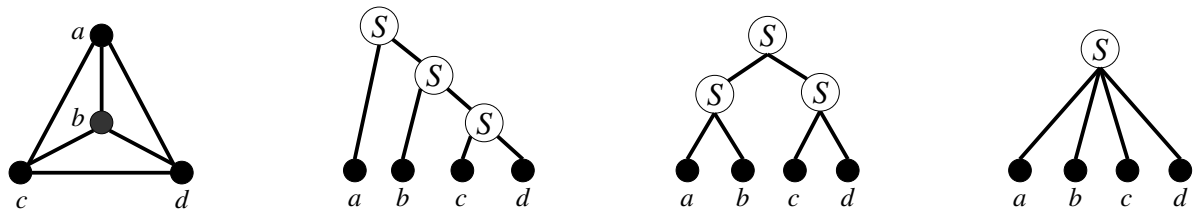
Les familles faiblement partitives sont une généralisation des familles partitives. Les modules d'un graphe orienté forment une famille faiblement partitive.

DÉFINITION 1.21 (Arbre de décomposition modulaire). Soit $G = (V, E)$ un graphe. L'arbre de décomposition modulaire T de G est un arbre enraciné où, les feuilles correspondent aux sommets V de G (bijection). Les noeuds internes sont étiquetés soit premier, soit séries ou enfin parallèles. Les noeuds séries et parallèles sont des noeuds complets, cela signifie que toute combinaison des enfants (comprendre l'union) du noeuds considéré forme un module. Les noeuds séries signifient que chaque enfant du noeud est complètement relié aux autres enfants. Par opposition, parallèles signifient que chaque enfant est complètement déconnecté de tous les autres. Les noeuds premiers ont la caractéristique suivante, chacun de ses enfant forme un module fort. Toute combinaison d'enfant d'un noeud premier n'engendre pas de modules.

Un graphe et son arbre de décomposition est présenté en figure 1.12.

La définition précédent laisse entendre que dans l'arbre de décomposition modulaire, nous pouvons avoir trois types de noeuds internes, et que les feuilles représentent les sommets du graphes. Ce qui permet pour un même graphe d'obtenir des arbre de décompositions modulaires différents, en effet en figure 1.11 nous avons une clique à 4 sommets, et plusieurs arbres possibles de décompositions modulaire associés à ce graphe. Il est cependant possible de remédier à ce problème en utilisant un arbre de décomposition canonique.

Définition 1.22 (Arbre Canonique de Décomposition Modulaire). Un arbre de décomposition modulaire est canonique si pour tout noeud parallèle, il n'a pas d'enfant parallèles. Et tout noeud série n'a pas d'enfant séries.



(a) K_4 (b) (c) (d)

Figure 1.11. Une clique à 4 sommets (a) et plusieurs arbres de décompositions modulaire de K_4 (b)-(d). Les arbres (b) et (c) ne sont pas canoniques, alors que (d) l'est.

Définition 1.23 (Graphes totalement décomposable). Soit $G = (V, E)$ un graphe, orient ou non, et soit $\mathbf{\Psi}$ une décomposition de graphe. G est dit complètement décomposable (de manière non triviale) vis à vis de $\mathbf{\Psi}$ si et seulement si G peut être décomposé par $\mathbf{\Psi}$ et tout sous-graphe induit H de G, de taille suffisante, admet également une décomposition non triviale.

Remarque 1.24. Dans la définition 1.23, la mention "taille suffisante" se réfère au graphe d'une certaine taille en dessous de laquelle les graphes sont tous complètement décomposables. Dans le cas de la décomposition modulaire, la taille critique est 3. En effet tous les graphes avec au plus trois sommets sont tous décomposables, alors qu'à partir de 4 sommets nous voyons apparaître des graphes que l'on ne peut pas décomposer : comme le P_4.

Une classe de graphe, nommée les cographes en référence aux *"Complement reducible graphs"* introduits par Corneil et al. (1981), correspond à la classe des graphes qui sont totalement décomposables à l'aide de la décomposition modulaire.

DÉFINITION 1.25 (Cographes Corneil et al. (1981)). Soit $G = (V, E)$ un graphe. Un cographe est un graphe défini de manière récursive :

(1) Un graphe à 1 sommet est un cographe.,

(2) Si $G_1, \ldots, G_k$ sont des cographes, alors $G = G_1 \cup \ldots \cup G_k$ est aussi un cographe,

(3) Si G est un cographe, alors $\overline{G}$ l'est également.

Les cographes admettent plusieurs caractérisations, en voici une liste non exhaustive,

PROPOSITION 1.26. *Soit $G = (V, E)$ un graphe. Les propositions suivantes sont équivalentes :*

(1) *G est un cographe.*

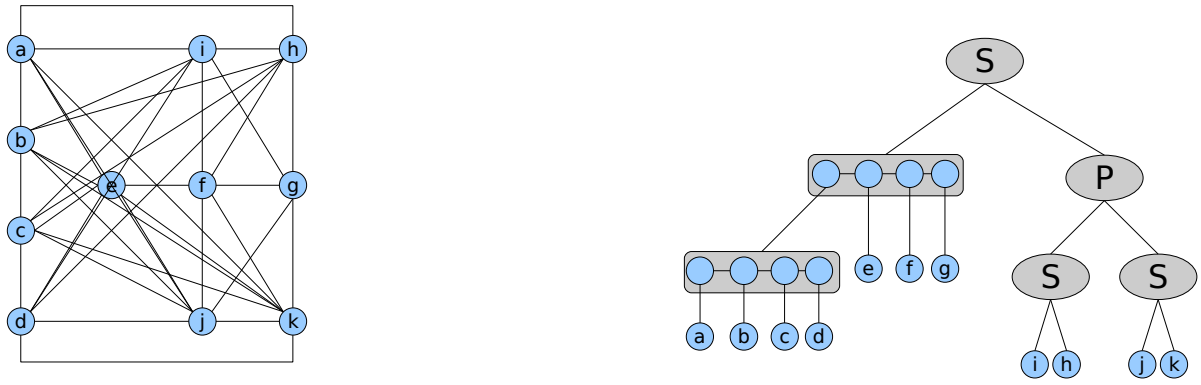
(2) *G ne contient pas P_4 comme sous graphe induit.*

(3) *L'arbre de décomposition modulaire de G contient seulement des noeuds séries et des noeuds parallèles comme noeuds internes.*

(4) *G peut être obtenu à partir d'un seul sommet par une séquence d'ajout de vrais et faux jumeaux.*

De nombreuses propriétés des cographes sont présentes dans les ouvrages de Brandstädt et al. (1999) ainsi que Golumbic (2004).

Il existe de nombreux algorithmes pour reconnaître cette classe de graphes, le premier algorithme linéaire est dû à Corneil et al. (1985), par la suite des algorithmes plus simples sont apparus Habib et Paul (2005) ; Bretscher et al. (2008).



(a) (b)

Figure 1.12. Un exemple de graphe (a) et son arbre de décomposition associé (b).

4.2. Algorithmes. La décomposition modulaire est un formidable outils théorique ayant de nombreuses applications, pourvu que la décomposition nous soit donnée. Heureusement pour nous, et contrairement à certaines décompositions, calculer la décomposition modulaire d'un graphe (*i.e.* trouver son arbre de décomposition) est un problème polynomial. Il existe même des algorithmes linéaires. Un des premiers algorithmes pour calculer la décomposition est apparu dans les années 70 par Cowan *et al.* [8]. Ce n'est que plus récemment que des algorithmes linéaires sont apparus. Indépendamment Cournier et Habib (1994) et McConnell et Spinrad (1994) ont été les premiers à proposer des algorithmes linéaires. Le principal défaut de ces algorithmes est qu'ils sont compliqués à implémenter. Par la suite plusieurs algorithmes linéaires plus simples sont apparus McConnell (1995) ; Dahlhaus (1995) ; Dahlhaus et al. (1997) ; McConnell et Spinrad (1999).

Pour calculer la décomposition modulaire deux grandes techniques algorithmiques se dégagent, la première consiste à trouver une branche gauche (*cf.* paragraphe suivant), cette technique sera d'ailleursutilisée au Chapitre. 2. La seconde technique consiste à calculer une permutation factorisante (également utilisée au Chapitre 2).

Finalement Tedder et al. (2008) ont proposé une approche hybride qui calcule la décomposition modulaire en calculant une permutation factorisante.

4.3. L'algorithme de Ehrenfeucht et al. (1994). En 1994 Ehrenfeucht, Gabow, McConnell et Sullivan ont présenté un algorithme de décomposition modulaire des 2-structures (*cf.* définition 1.27) en temps $O(n^2)$. Ce qui, dans le cas des 2-structures, constitue un algorithme linéaire.

Mais dans le cas des graphes, c'est algorithme est loin d'être linéaire. Cependant, il est intéressant dans la mesure où la technique présentée a été utilisée de nombreuses fois pour concevoir des algorithmes de décompositions modulaire efficaces. Nous pourrons retenir les algorithmes de McConnell et Spinrad (2000) et de Dahlhaus et al. (2001) comme des adaptations de l'algorithme de Ehrenfeucht et al. (1994). Ils admettent comme complexité pour Dahlhaus et al. (2001) $O(n + m.\alpha(n, m))$ et ensuite $O(n + m)$ mais difficile à implémenter et pour McConnell et Spinrad (2000) $O(n + m\ log\ n)$, qui utilise l'affinage de partition.

Le principe de l'algorithme est le suivant, il faut tout d'abord sélectionner un sommet x, et calculer ce que l'on appelle la branche$-x$. Il s'agit en quelque sorte de l'épine dorsale de l'arbre de décomposition modulaire en partant de x. Cette branche$-x$ contient tous les modules forts qui contiennent x. Et d'après la définition 1.15, la branche$-x$ est une chaîne d'inclusions. Ensuite il faut calculer les modules maximaux (vis à vis de l'inclusion) qui

[8]Un historique plus détaillé se trouver à l'adresse suivante : `http://www.liafa.fr/~fm/HistDM.html` et une version documentée par P. Gambette à l'adresse suivante : `http://www.lirmm.fr/~gambette/EnsDecompositionModulaire.php`

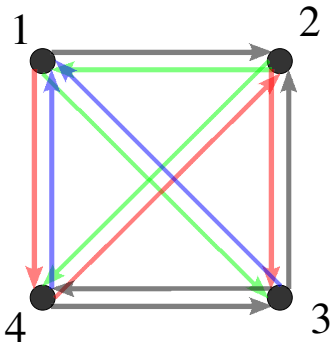
ne contiennent pas x. Une fois que cela est effectué il *"suffit"* de lancer la récursion sur les modules non triviaux, *i.e.* dans chaque module fort qui ne contient pas x, nous prenons un sommet y et reproduisons le processus. Une présentation plus détaillée est présentée dans l'habilitation de Paul (2006).

Nous montrerons au Chapitre 2 comment généraliser cet algorithme pour calculer la décomposition modulaire sur les *bonnes* relations homogènes, tout en conservant la même complexité, à savoir linéaire sur la structure que nous considérerons. Une structure combinatoire qui sera introduite dans la suite du document.

DÉFINITION 1.27 (2-Structure Ehrenfeucht et Rozenberg (1990a,b,c) ; Ehrenfeucht et al. (1999)).

Soit $S = (D, R)$ une 2-structure où D est un ensemble fini, R est une relation d'équivalence sur $E_2(D)$ antisymétrique.

En d'autres termes, une 2-structure peut être vu comme une clique (orienté symétrique) où tous les arcs ont une couleur.



(a)

La 2-structure $g = (D, \mathcal{P})$ où
$D = \{1, 2, 3, 4\}$ et
$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4\}$

$\mathcal{P}_1 = \{(1,2), (3,2), (3,4), (4,3)\}$,
$\mathcal{P}_2 = \{(1,3), (2,1), (2,4)\}$,
$\mathcal{P}_3 = \{(1,4), (2,3), (4,2)\}$,
$\mathcal{P}_4 = \{(3,1), (4,1)\}$.

(b) $g = (D, \mathcal{P})$

Figure 1.13. Une 2-structure sous forme graphique (a) et sa définition en en extension (b)

5. Plan

Le document se présente en deux parties. La première partie (Chapitres. 2 et 3) étudie les généralisations de la décomposition modulaire. La seconde partie (Chapitres 4 et 5) s'intéresse à la conception d'algorithmes efficaces.

Le Chapitre. 2 concerne les *Relations Homogènes*. Ol s'agit d'une nouvelle structure combinatoire, qui entre autre chose généralise les graphes.

Les relations homogènes ont été introduite afin de fournir un cadre général à la décomposition modulaire tout en conservant les *bonnes* propriétés. À savoir les propriétés structurelles fortes, une complexité polynomiale pour trouver une décomposition. Cette généralisation s'appuie sur le concept fondamental de *distinction*, pour ne retenir comme principe directeur qu'un module d'une relation homogène, est un ensemble perçu uniformément par l'extérieur.

Dans cette première partie nous présenterons les relations homogènes, nous fournirons une définition formelle et présenterons quelques premières propriétés intéressantes de ces relations. Dans un premier temps nous considérons les problèmes sur des relations homogènes assez particulières et nous fournirons, sur ces dernières, des algorithmes efficaces. Nous présenterons ensuite des solutions algorithmiques sur des relations homogènes générales. Enfin nous montrerons en quoi ce nouveau concept de relations homogènes, permet de revenir sur les graphes avec une notion de module différente de la définition originale proposée par Gallai (1967), et ce sans que les principales propriétés de la décomposition modulaire en soient altérées.

Tout en proposant une abstraction de la décomposition modulaire par le biais des relations homogènes, nous sommes tout de même capable de présenter des algorithmes admettant des complexités linéaires, ou quasi linéaires.

Le Chapitre. 3 est dédié à la présentation et l'étude d'une nouvelle décomposition : la décomposition *Umodulaire*. Le concept d'umodule vise, en quelque sorte, à généraliser la notion de module. Les umodules sont définis sur les relations homogènes, introduites en première partie. Comparativement aux modules, il s'agit de changer de point de vue. En effet dans le cas des modules, c'est l'extérieur qui perçoit le module de manière uniforme alors que dans le cas des umodules, les rôles sont inversés : les umodules, distinguent l'extérieur de la même façon. Dans un premier temps, nous présenterons formellement le concept d'umodules, nous montrerons les premières propriétés combinatoires intéressantes que nous avons obtenu sur cette décomposition.

Nous étudierons également le problème fondamental qu'est le calcul de cette décomposition. Lorsque bien sûr, c'est possible. C'est pourquoi nous considérerons des relations homogènes équipées de propriétés supplémentaires. Nous verrons ensuite à quoi correspondent les umodules lorsque l'on considère les graphes comme structure. Nous verrons ce qu'il en est pour les graphes non-orientés. Et nous montrerons au passage comment une opération de transformation sur la relation homogène, nommée *Seidel Switch*, permet dans certains cas (en particulier les graphes non-orientés) de se ramener à un problème de décomposition modulaire. Par la suite nous appliquerons cette décomposition à une structure

proche des graphes non-orientés[9], à savoir les tournois. Sur ces derniers, la décomposition umodulaire constitue une décomposition complètement nouvelle. Par la suite, nous caractériserons la classe des tournois qui sont complètement décomposables vis à vis de cette décomposition. Et nous présenterons, sur la classe des tournois complètement décomposables, deux algorithmes linéaires pour résoudre deux problèmes difficiles sur les tournois en général. À savoir le problème de l'isomorphisme de tournois et le problème du *Feedback Vertex Set* [10].

Le Chapitre. 4 est consacré au problème du calcul des composantes de chevauchement d'une famille de parties. Les problème consiste, étant donné une famille de parties d'un ensemble fini, de calculer les composantes de chevauchement. Bien qu'en apparence ce problème semble assez éloigné des deux chapitres précédents, en réalité ce n'est pas complètement le cas. En effet, le premier à avoir donné un solution efficace à ce problème fût Dahlhaus. Ce problème intervenait comme une sous-procédure pour le calcul efficace de la décomposition en coupes. Ce problème intervient également dans la première partie de ce document (Chapitre. 2 section. 4). L'algorithme que nous allons présenter est une simplification et en un sens une amélioration de l'algorithme de Dahlhaus. Le papier dans lequel fût publié cet algorithme est écrit d'une manière pour le moins confuse, et en l'état ne fournissait pas toutes les garanties de correction de l'algorithme. Dans ce chapitre nous présentons la version originale de Dahlhaus, et donnons une preuve de correction. Par la suite nous présentons une version simplifiée qui se débarrasse de toute une partie algorithmique lourde et complexe à implémenter à savoir des calculs du plus petit ancêtre commun. Et nous remplaçons cette procédure par une technique algorithmique plus légère, à savoir l'affinage de partition. Enfin, nous montrons comment modifier notre algorithme pour obtenir directement un sous-graphe couvrant du graphe de chevauchement.

Enfin, le Chapitre. 5 présente un algorithme efficace pour reconnaître les graphes qui ont une largeur NLC[11] 2. Nous améliorons un résultat de Johansson, et présentons une décomposition canonique des graphes de largeur NLC égale à 2, permettant au passage de traiter le problème de l'isomorphisme des graphes de largeur NLC 2.

[9]Dans le sens où sur les tournois comme sur les graphes non orientés, il y a seulement deux types de voisinages

[10]En Français : plus petit ensemble de sommets retours ou encore ensemble coupe circuit

[11]Node Labelled Controlled.

Les travaux menés lors de cette thèse ont donné lieu aux publications suivantes :

2006

(1) Binh-Minh Bui-Xuan, Michel Habib, Vincent Limouzy, and Fabien de Montgolfier, *Homogeneity vs. adjacency : generalising some graph decomposition algorithms*, 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG) (Fedor V. Fomin, ed.), LNCS, vol. 4271, Springer, June 2006.

(2) ———, *On modular decomposition concepts : the case for homogeneous relations*, Electronic Notes in Discrete Mathematics **27** (2006), 13–14.

2007

(3) ———, *Unifying two graph decompositions with modular decomposition*, Algorithms and Computation, 18th International Symposium, ISAAC 2007, Sendai, Japan, December 17-19, 2007 (Takeshi Tokuyama, ed.), LNCS, vol. 4835, Springer, December 2007.

(4) Vincent Limouzy, Fabien de Montgolfier, and Michaël Rao, *NLC-2 graph recognition and isomorphism*, 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG), LNCS, vol. 4769, Springer, June, pp. 86–98. 2007

2008

(5) Binh-Minh Bui-Xuan, Michel Habib, Vincent Limouzy, and Fabien de Montgolfier, *Algorithmic aspects of a general modular decomposition theory*, Discrete Applied Mathematics, à Paraître (2008).

(6) ———, *A new tractable combinatorial decomposition.*, soumis (2008).

(7) Pierre Charbit, Michel Habib, Vincent Limouzy, Fabien de Montgolfier and Mathieu Raffinot, *A note on computing set overlap classes*, Information Processing Letters (2008), à Paraître.

Première partie

Décomposition Modulaire et Généralisations.

CHAPITRE 2

Relations homogènes et décomposition modulaire

Nous introduisons dans ce chapitre les *Relations Homogènes*. Les relations homogènes constituent une nouvelle structure combinatoire, elles ont été introduites afin d'abstraire le concept de *modules* dans les graphes.

Rappelons qu'un module est un ensemble de sommets d'un graphe qui est uniformément perçu par le "monde" extérieur. Il s'avère donc que la notion de distinction est une notion fondamentale pour l'étude des modules. Finalement un module n'est rien d'autre qu'un ensemble de sommets qui n'est pas distingué par le reste des sommets.
En quelques mots, une relation homogène est une structure discrète définie sur un ensemble fini, et où à chaque élément de l'ensemble support nous associons des classes d'équivalence de l'ensemble support privé de l'élément considéré. En d'autres termes un élément x ne distingue pas deux éléments y et z si et seulement si ces derniers appartiennent à une même classe d'équivalence associée à x. Nous pouvons donc définir une notion de module sur cette structure comme étant un ensemble de sommet non *"distingués"* par les éléments extérieurs.

En définitive, par le biais des relations homogènes nous fournissons une abstraction du concept de module, s'affranchissant ainsi de la structure et de la relation d'adjacence classique pour ne conserver seulement la notion essentielle de distinction. Cela nous permet de fournir un cadre général où l'on peut étudier la décomposition modulaire, et mieux comprendre quelles sont les conditions nécessaires pour que sur certaines structures le calcul de cette dernière soit facilité. De plus ces relations homogènes fournissent une généralisation des graphes et même des structures plus générales que sont les 2-structures.

Ce chapitre est organisé de la façon suivante : Nous commençons (Section. 1) par définir formellement les relations homogènes, et nous donnons quelques exemples de relations homogènes afin que le lecteur puisse se familiariser avec cette nouvelle notion. Nous considérons ensuite la question essentielle de la représentation de ces relations homogènes. En effet d'un point de vue algorithmique, nous devons être capable de représenter ces dernières de manière raisonnable si ce n'est efficace. Ensuite nous abordons la décomposition modulaire (Section. 2) des relations homogènes. Nous montrons S comment récupérer les graphes sous ce formalisme. Nous étudions ensuite des relations homogènes particulières, ce sont des relations homogènes qui satisfont plus de propriétés que les propriétés standards

de la définition 2.1. Nous faisons ensuite le lien entre les famille partitives et les relations homogènes (Section. 3). Ensuite nous nous intéressons aux questions algorithmiques, et plus précisément tester la primalité de la relation homogène et calculer la décomposition modulaire lorsque cela est possible. Dans un premier temps (Section. 4) nous considérons les relations homogènes arbitraires. Nous obtenons un algorithme de décomposition modulaire qui s'exécute en temps $O(n^3)$. Dans un second temps nous verrons comment sur les *bonnes* relations homogènes il est possible, grâce aux propriétés supplémentaires de ces relations, d'obtenir de meilleurs résultats en termes de complexité. Nous présenterons, pour ces relations, un algorithme de décomposition en temps $O(n^2)$. Et finalement nous présentons (Section 6) des perspectives sur les relations homogènes et nous montrons, grâce aux relations homogènes, comment il est possible de définir sur les graphes d'autres types de décomposition modulaire, différentes de la définition originale.

Les résultats présentés dans ce chapitre ont donné lieu aux publications suivantes : Bui-Xuan et al. (2006b), Bui-Xuan et al. (2006a) et Bui-Xuan et al. (2008a).

Plan

1. Définitions et représentations

DÉFINITION 2.1 (Relation homogène[1]). Soit X un ensemble fini. Une relation homogène H est définie pour tout triplet a, b, c de X, on note $H(a|bc)$.

La relation H satisfait les conditions suivantes :

(1) **Réflexivité :** $H(a|bb)$.

(2) **Symétrie :** $H(a|bc) = H(a|cb)$.

(3) **Transitivité :** $H(a|bc)$ et $H(a|cd) \Rightarrow H(a|bd)$.

On dit que a est homogène vis à vis de b et c.

Par la suite nous prendrons comme convention que $n = |X|$, sauf mention contraire. Nous considérerons également que les relations homogènes utilisées sont définies sur un ensemble fini.

[1] Il s'agit en quelque sorte d'un retour au source, car au début les modules étaient appelés des *"parties homogènes"* qui était la traduction de *"externally related set"*.

REMARQUE 2.2 (Représentation des relations homogènes). Bien que les relations homogènes soient définies comme des ensembles de triplets, il est toutefois possible de les stocker et les manipuler en un espace moindre que le $O(n^3)$ que laisse supposer la définition. En effet vu que les relations homogènes satisfont plusieurs conditions assez strictes (*cf.* définition 2.1). Il est possible d'associer à chaque élément x de X une partition de $X \setminus \{x\}$. et par conséquent nous obtenons un stockage de taille $O(n^2)$. Soit sous forme de matrice, soit sous forme de famille de partitions.

La partition associée à un élément x de X est notée : $H_x = \{H_x^1, \ldots, H_x^k\}$

NOTATION 2.3 (Relation Homogène Induite). Soit H une relation homogène sur l'ensemble X. Soit A un sous-ensemble de X, on notera $H[A]$ la relation homogène restreinte aux éléments de A.

DÉFINITION 2.4 (Congruence Locale). La congruence locale d'un relation homogène H définie sur l'ensemble X est le nombre maximum de classes que comporte la partition H_x associée à l'élément x. Plus formellement la congruence locale (CL) de H est donnée par :

$$CL(H) = \max_{x \in X} \{|H_x|\}$$

DÉFINITION 2.5 (Relation Standard d'un Graphe). Soit H une relation homogène sur l'ensemble X, on dit que la relation homogène est graphique s'il existe un graphe $G = (X, E)$ de telle sorte que la relation H' obtenue à partir de G en associant pour chaque sommet v de G la partition voisinage et non-voisinage de x, *i.e.* $\{N(x), \overline{N(x)}\}$.

Un exemple de relation homogène est donnée en figure 2.1.

2. Décomposition modulaire

Dans cette section nous considérons la décomposition modulaire des relations homogènes, nous montrerons quelques unes de ses propriétés, en particulier où résident les différences entre la décomposition modulaire des graphe et la décomposition modulaire des relations homogènes. Pour cela commençons par définir les modules sur cette structure.

DÉFINITION 2.6 (Module d'une Relation Homogène). Soit X un ensemble fini, et H une relation homogène sur X. Un module de H est un sous-ensemble M de X tel que pour tout élément x appartenant à $X \setminus M$ et pour tout couple m, m' de M on ait :

$$H(x|mm')$$

REMARQUE 2.7 (Modules Triviaux). Nous remarquerons que quelle que soit la relation homogène considérée il existe toujours des modules. En effet l'ensemble X est un module, par définition, il est perçu de la même façon par les éléments à l'extérieur, puisqu'il n'y

$$H(a|bc),\ H(a|bd),\ H(a|be),\ H(a|bf),\ H(a|cd),\ H(a|ce),\ H(a|cf),$$
$$H(a|de),\ H(a|df),\ H(a|ef), H(b|ae),\ H(b|cc),\ H(b|df),\ H(c|ad),$$
$$H(c|be),\ H(c|ef),\ H(c|bf),\ \cdots$$

(a) Représentation en extension.

$$\begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 2 & 3 & 1 & 3 \\ 1 & 2 & 0 & 1 & 2 & 2 \\ 1 & 2 & 3 & 0 & 4 & 5 \\ 1 & 1 & 1 & 2 & 0 & 2 \\ 2 & 1 & 2 & 1 & 1 & 0 \end{pmatrix}$$

(b) Relations homogène sous forme matricielle.

$$\begin{array}{ll} a & \{b,c,d,e,f\} \\ b & \{a,e\},\{c\},\{d,f\} \\ c & \{a,d\},\{b,e,f\} \\ d & \{a\},\{b\},\{c\},\{e\},\{f\} \\ e & \{a,b,c\},\{d\},\{f\} \\ f & \{b,d\},\{a,c,e\} \end{array}$$

(c) Relation Homogène sous forme de "listes".

Figure 2.1. Exemple d'une relation homogène définie sur $X = \{a, b, c, d, e, f\}$, définie en extension (a), puis sous forme de matrice (b) et enfin sous forme de listes (c)

en a aucun. Les singletons $\{x\}$ sont aussi des modules. Ces deux familles de modules sont appelés les modules triviaux, car ils sont tout le temps présents. Les modules de taille supérieure à 1 et inférieure à n sont dits non triviaux.

NOTATION 2.8 (Primalité). Une relation homogène H sur l'ensemble X est dite première si tous ses modules sont triviaux.

DÉFINITION 2.9 (Casseur). Soient H une relation homogène définie sur l'ensemble X et A un sous-ensemble de X. Un élément x de $X \setminus A$ est un casseur de A s'il existe $a, b \in A$ tel que $\overline{H(x|ab)}$. En d'autres termes x empêche que l'ensemble A soit un module.

REMARQUE 2.10 (Dualité Module Casseur.). Pour une relation homogène H définie sur X, un sous-ensemble M est un module de H si et seulement si $X \setminus M$ ne contient pas de casseur.

NOTATION 2.11 (Famille des Modules). Soit H une relation homogène définie sur l'ensemble X. On notera $\mathscr{M}_H$ la famille des modules de la relation homogène H.
plus formellement :

$$\mathscr{M}_H = \{M | M \ \subseteq X \text{ et } M \text{ module de } \ H\}$$

Voici un première propriété importante des modules sur les relations homogènes.

LEMME 2.12. *Soit H une relation homogène définie sur X. Soient A et B deux modules de H tel que $A \mathbin{@} B$, alors*

$$A \cup B \in \mathscr{M}_H \text{ et } A \cap B \in \mathscr{M}_H$$

DÉMONSTRATION. Il est facile de voir que $A \cap B$ est un module, en effet, pour tout x à l'extérieur de $A \cap B$ nous avons $H(x|cd)$ pour c et d des éléments de l'intersection de A et B.

En ce qui concerne l'union, ce n'est pas plus difficile. En effet considérons x un éléments de $X \setminus (A \cup B)$ et soient a un élément propre de A, b un élément propre de B et c un élément de $A \cap B$. Comme A et B sont des modules, nous avons par définition $H(x|ac)$ et $H(x|bc)$, en utilisant maintenant la transitivité de H_x : nous obtenons $H(x|ab)$ et nous en déduisons que $A \cup B$ est également un module. □

Remarquons que contrairement aux modules des graphes, nous ne pouvons rien dire sur la différence de deux modules qui se chevauchent.

DÉFINITION 2.13 (Famille Ring/Lattice). Soit X un ensemble fini, et $\mathscr{F}$ une famille de sous-ensembles de X. Une famille $\mathscr{F}$ est dite *"Ring"* ou treillis, si pour toute paire d'éléments A et B de la famille alors

$$A \cup B \in \mathscr{F} \text{ et } A \cap B \in \mathscr{F}$$

Ces familles "Ring" apparaissent naturellement en combinatoire, par conséquent le problème du codage compact de la famille est un problème important. Un codage compact a été proposé par Grötschel et al. (1988) il est en espace $O(n^2)$. Il s'avère que cette borne est optimale, Bernàth (2004) a exhibé une famille pour laquelle la borne est atteinte.

Bien que ce type de famille soit intéressante en combinatoire, elle ne nous est d'aucune aide pour coder de manière efficace la famille des modules d'une relation homogène. En revanche, il existe un type de famille plus générale que les familles *"Ring"* qui sont les familles intersectantes. Ces dernières sont plus adaptées pour coder les familles des modules.

DÉFINITION 2.14 (Famille Intersectante). Soit X un ensemble fini, et $\mathscr{F}$ une famille sur 2^X. Une famille $\mathscr{F}$ est intersectante (Intersecting Grötschel et al. (1988) ; Gabow (1993, 1995) ; Bui-Xuan (2008)) si pour tout $A, B \in \mathscr{F}$ tel que $A \otimes B$ on ait :

$$A \cup B \in \mathscr{F} \text{ et } A \cap B \in \mathscr{F}$$

PROPOSITION 2.15. *Soit H une relation homogène définie sur l'ensemble X. La famille des modules $\mathscr{M}_H$ forme un famille intersectante. Il est possible de coder cette famille intersectante en espace $O(n^2)$ où n est la cardinalité de l'ensemble X.*

DÉMONSTRATION. En ce qui concerne le premier point, la preuve est donnée par le lemme 2.12. Quant au second point Gabow (1993, 1995) donne une représentation sous forme d'arbres des familles intersectantes en espace $O(n^2)$ (la méthode employée est assez technique et ne sera pas détaillée ici). □

2.1. Modules et fonctions sous-modulaires. Les fonctions sous-modulaires jouent un rôle fondamental en optimisation combinatoire, elles fournissent une structure sous-jacente à la structure considérée. Et elles permettent en général de résoudre efficacement les problèmes rencontrés.

Les fonctions sous-modulaires ont commencé à être étudiées dans les années quarante, entre autres par Rado (1942) qui considérait le rang de matroïdes puis vint Ingleton (1959) ou encore Rado (1957) et concernant l'optimisation combinatoire Edmonds (1970).

Plus de précisions et d'applications des fonctions sous-modulaires sont présentés dans l'ouvrage de référence sur le sujet Fujishige (1991), ainsi que dans le livre de (Grötschel et al., 1988, *"Chapter 10: Submodular Functions"*) et également dans l'oeuvre de Schrijver (2003).

Les fonctions sous-modulaires ont connu un regain d'intérêt depuis que Schrijver (2000) et Iwata et al. (2000, 2001) ont montré que le problème de minimiser une fonction sous-modulaire est un problème fortement polynomial.

Ces résultats on depuix été amélioré par Iwata (2002) et Iwata (2003), ainsi que Vygen (2003). plus récemment Orlin (2007) a encore abaissé la complexité. Un synthèse de tous ces algorithmes de minimisation peut être trouvé dans les travaux de Iwata (2008).

DÉFINITION 2.16 (Fonction sous-modulaire). Soit X un ensemble (possiblement fini), f est une fonction de $2^X \to \mathbb{R}$. f est dite sous-modulaire si pour tout sous-ensemble A et B de X on a :

$$f(A) + f(B) \geqslant f(A \cup B) + f(A \cap B)$$

THÉORÈME 2.17 (Casseurs et Sous-Modularité). *Soit H une relation homogène définie sur l'ensemble X. Et soit s la fonction qui associe à un sous-ensemble A de X le nombre de casseurs de A ($s : 2^X \to \mathbb{N}$). Alors cette fonction respecte l'inégalité sous-modulaire pour toutes les paires d'ensembles intersectants :*

$$\forall A, B \subseteq X \text{ tel que } A \cap B \neq \varnothing$$

$$s(A) + s(B) \geqslant s(A \cup B) + s(A \cap B),$$

DÉMONSTRATION.

Si $A \subset B$ ou $B \subset A$, le résultat est évident.

Considérons maintenant le cas où $A \circledcirc B$. $\mathcal{S}_A$ désigne l'ensemble des casseurs de A. $\mathcal{S}_{A\cap B} = (\mathcal{S}_{A\cap B} \setminus B) \uplus (\mathcal{S}_{A\cap B} \cap B)$.

Nous avons $\mathcal{S}_A \cap A = \varnothing$.

$$\begin{aligned}\mathcal{S}_{A\cup B} &= (\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A) \uplus (\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A) \\ &= (\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A) \uplus (\mathcal{S}_A \setminus (A \cup B))\end{aligned}$$

De manière similaire $\mathcal{S}_B = (\mathcal{S}_B \setminus \mathcal{S}_{A\cap B}) \uplus (\mathcal{S}_{A\cap B} \setminus B)$.
Et enfin

$$\begin{aligned}\mathcal{S}_A &= (\mathcal{S}_A \setminus B) \uplus ((\mathcal{S}_A \cap B) \setminus \mathcal{S}_{A\cap B}) \uplus ((\mathcal{S}_A \cap B) \cap \mathcal{S}_{A\cap B}) \\ &= (\mathcal{S}_A \setminus (A \cup B)) \uplus ((\mathcal{S}_A \cap B) \setminus \mathcal{S}_{A\cap B}) \uplus B \cap \mathcal{S}_{A\cap B}\end{aligned}$$

Nous obtenons :

$$|\mathcal{S}_A| + |\mathcal{S}_B| - |\mathcal{S}_{A\cup B}| - |\mathcal{S}_{A\cap B}| = |(\mathcal{S}_A \cap B) \setminus \mathcal{S}_{A\cap B}| + |\mathcal{S}_B \setminus \mathcal{S}_{A\cap B}| - |\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A|$$

Pour compléter la preuve nous devons montrer que $\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A$ est inclus dans $\mathcal{S}_B \setminus \mathcal{S}_{A\cap B}$. Considérons pour cela un élément s de $\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A$, par conséquent $s \notin A \cup B$ et nous avons $H(s|xy)$ pour toute paire d'éléments x, y de A. Supposons maintenant que $s \notin \mathcal{S}_B$. Comme s n'appartient pas à B on déduit donc que l'on a $H(s|xy)$ pour toute paire d'éléments x, y de B. De plus nous rappelons que A et B se chevauchent, par conséquent de la propriété de transitivité nous déduisons que nous avons $H(s|xy)$ pour tout x et y appartenant à $A \cup B$ et $s \notin A \cup B$ et donc $s \notin \mathcal{S}_{A\cup B}$. Donc contradiction. Finalement, si nous supposons que s appartient à $\mathcal{S}_{A\cap B}$ cela implique que s appartient à $\mathcal{S}_A$. □

Une version plus faible du théorème précédent à été publiée par Uno et Yagiura (2000), ils ont proposé un algorithme élégant pour calculer les intervalles communs à deux permutations, pour ce faire ils s'appuient fortement sur la propriété de sous-modularité.

Cette approche a ensuite été généralisée par Bui-Xuan et al. (2005) pour les modules d'un graphe non orienté.

Qu'en est il pour les relations homogènes arbitraires ?

2.2. Modules forts et primalité.

Dans une famille $\mathscr{F}$ de sous-ensembles de X quelconque, un membre A de la famille est dit *fort* si il n'existe aucun autre élément B de la famille tel que $A \oslash B$. Tous les membres de $\mathscr{F}$ qui ne sont pas *forts* sont dits *faibles*. Si X et $\{x\}$ appartiennent à $\mathscr{F}$, sont appelés les membres triviaux (et forts) de la famille (triviaux, car personne ne peut les chevaucher). Si la famille ne contient pas les membres triviaux, nous étendons $\mathscr{F}$ en ajoutant les membres triviaux forts.

Les membres forts de la famille sont ordonnés par inclusion, l'ordre obtenu forme un arbre (nous rappelons que pour toute paire de membres forts soit l'un est inclus dans l'autre, soit ils sont disjoints). Par la suite, cet arbre sera appelé *l'arbre de décomposition modulaire généralisé*.

DÉFINITION 2.18 (Famille Laminaire). Soit X un ensemble fini, et soit $\mathscr{F}$ une famille de sous-ensembles de X. $\mathscr{F}$ est une famille laminaire si pour toute paire d'éléments A, B de $\mathscr{F}$, exactement une des condition suivante est vérifiée :

- $A \subset B$.
- $B \subset A$.
- $A \cap B = \varnothing$.

PROPOSITION 2.19 (Folklore). *Il est possible de représenter une famille laminaire $\mathscr{F}$ définie sur l'ensemble X, en espace $O(n)$.*

Une preuve de la proposition précédente peut être trouvée dans Edmonds et Giles (1977) ou Schrijver (2003).

Cela constitue une preuve simple qu'une famille $\mathscr{F}$ sur X comporte au plus $2n-1$ (où $n = |X|$) membres forts et au plus $n-2$ membres non triviaux, comme l'arbre a exactement n feuilles, et aucun noeud interne de degré 2 sauf éventuellement pour la racine.

Lorsque la famille $\mathscr{F}$ considérée est *faiblement partitive*, cet arbre joue un rôle très important (fondamental) car il constitue un codage exact en espace $O(n)$ de la famille $\mathscr{F}$. Alors que cette famille peut comporter jusqu'à $O(2^n)$ membres. Nous l'appellerons alors *l'arbre de décomposition* de $\mathscr{F}$.

Le parent, dans l'arbre de décomposition, d'un membre M de $\mathscr{F}$, éventuellement faible, est le plus petit membre fort M_P qui contient M, M est alors appelé fils de M_P.

Par exemple si M est un élément fort, M_P est son parent dans *l'arbre généralisé de décomposition.*

Un élément fort est *premier* si tous ses enfants sont forts, et *complet* dans le cas contraire.

Une *classe de chevauchement* de $\mathscr{F}$ est une relation d'équivalence de la fermeture transitive de la relation de chevauchement $\owedge$ sur $\mathscr{F}$ (*cf.* figure 2.2). Pour plus de détails sur le problème on pourra se référer au Chapitre. 4 qui est consacré au calcul de ces composantes.

Une classe de chevauchement est *triviale* si elle contient un seul élément A de $\mathscr{F}$. En conséquence de quoi, A est un élément fort de $\mathscr{F}$.

Le *support* d'une classe de chevauchement $\mathcal{C} = \{C_1, \ldots, C_k\}$ est défini comme $S(\mathcal{C}) = C_1 \cup \ldots \cup C_k$.

Un *atome* d'une classe de chevauchement $\mathcal{C}$ est un sous-ensemble maximal de $S(\mathcal{C})$ qui ne chevauche aucun des C_i, avec $1 \leqslant i \leqslant k$ (une illustration est donnée en figure 2.2). Nous remarquerons que les atomes de $\mathcal{C}$ forment une partition de $S(\mathcal{C})$. De plus un atome d'une classe de chevauchement appartient à la classe si et seulement si cette classe est triviale. Qui plus est le support, respectivement un atome, d'une classe de chevauchement appartient à la famille $\mathscr{F}$ si et seulement si il s'agit d'un élément fort de $\mathscr{F}$.

Il est clair que le support ou un atome d'une classe de chevauchement n'appartient pas forcément à la famille $\mathscr{F}$. Cependant dans le cas des familles faiblement partitives, tous les atomes et tous les supports des classes de chevauchements appartiennent à la famille, par la définition même de la partitivité de $\mathscr{F}$, et par conséquent ils constituent les membres forts de la famille.

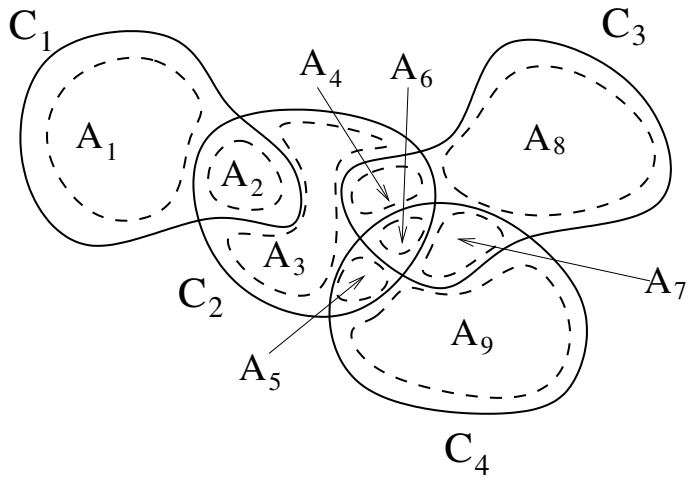


Figure 2.2. Les atomes :$A_1, A_2, \ldots, A_9$ de la classe de chevauchement $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$

PROPOSITION 2.20. *Ce qui suit est vrai pour toute famille $\mathscr{F}$ de sous-ensembles d'un ensemble fini X, qui est fermé par union d'éléments se chevauchant.*

(1) *$A \subseteq X$ est un élément premier et fort de $\mathscr{F}$ si et seulement si $\{A\}$ est une classe de chevauchement triviale.*

(2) *$A \subseteq X$ est un élément complet et fort de $\mathscr{F}$ si et seulement si c'est le support d'au moins une classe de chevauchement non-triviale $\mathcal{C}_A$ de $\mathscr{F}$. Dans ce cas les, enfants faibles de A coïncident avec les membres de $\mathcal{C}_A$.*

Nous appliquons maintenant ces notions à la famille des modules d'une relation homogène H. Considérons $Z(x, y)$ comme étant le plus gros module de H contenant x mais pas y. Nous sommes convaincus que $Z(x, y)$ est bien défini, en effet comme il s'agit de l'union de modules de la relation homogène H qui contiennent x mais pas y qui est un module de H en accord avec la proposition 2.20.

De plus $Z(x, y)$ n'est pas vide car nous sommes au moins sûrs que $\{x\}$ est un membre de la famille.

Soit $\mathcal{Z}(H)$ la famille :

$$\mathcal{Z}(H) = \{Z(x, y) | x, y \in X \text{ et } x \neq y\}$$

Nous remarquerons que $\mathcal{Z}(H)$ n'est pas nécessairement close par l'union d'éléments qui se chevauchent. Pour s'en convaincre, considérons l'exemple suivant :

$X = \{a, b, c\}$, avec $H(a|bc)$, $H(b|ac)$ et $H(c|ab)$ nous en déduisons que $\{a, b\} \in \mathcal{Z}(H)$, $\{a, c\} \in \mathcal{Z}(H)$ alors que X n'appartient pas à $\mathcal{Z}(H)$.

THÉORÈME 2.21. *Tous les supports et les atomes de $\mathcal{Z}(H)$ qui sont des modules de H sont des modules forts.*

Un module fort non trivial de H est soit le support ou un atome d'une classe de chevauchement de $\mathcal{Z}(H)$.

DÉMONSTRATION. Commençons par démontrer la première partie du théorème :

(1) Le support d'une classe de chevauchement de $\mathcal{Z}(H)$ est un module de H, car la famille des modules est close par union d'éléments qui se chevauchent (*cf.* proposition 2.20).

Si le support S d'une classe de chevauchement $\mathcal{C}$ est chevauché par un autre module, il est alors chevauché par un module $A \notin \mathcal{Z}(H)$. Considérons maintenant un élément x de $A \setminus S$ et y un élément de $S \setminus A$. Par conséquent $Z(x, y)$ contient A mais pas $\{y\}$ et donc chevauche S, il doit donc chevaucher au moins un membre de $\mathcal{C}$ et par conséquent $Z(x, y) \in \mathcal{C}$, une contradiction car $x \notin S$. En conclusion le support d'une classe de chevauchement est un module fort.

(2) Soit A un atome d'une classe de chevauchement $\mathcal{C}$ de $\mathcal{Z}(H)$. Si A est inclus dans au moins deux membres de $\mathcal{C}$, alors A est exactement l'intersection de tous les membres de $\mathcal{C}$ qui contiennent A. Et comme la famille des modules est fermée par intersection de membres chevauchants, (*cf.* proposition 2.20) A est un module. Nous remarquons que si A est inclus dans un seul élément de $\mathcal{C}$, A peut ne pas être un module.

Supposons que A soit un module de H, et qu'il soit chevauché par un autre module. Il est donc chevauché par un module $B \notin \mathcal{Z}(H)$. Soit x un élément de $B \setminus A$ et y un élément de $A \setminus B$. Il en découle que $Z(x, y)$ contient B mais pas $\{y\}$ et par conséquent chevauche A, donc il chevauche tous les éléments de $\mathcal{C}$ qui contiennent A et donc par conséquent $Z(x, y) \in \mathcal{C}$, ce qui est une contradiction car aucun atome, par la définition même d'un atome, ne peut être chevauché par un membre de la classe de chevauchement. Par conséquent les atomes d'une classe de chevauchement qui sont des modules, sont forts.

Montrons maintenant que si M est un module fort non trivial alors il est soit le support soit un atome d'une classe de chevauchement. Pour ce faire nous considérons trois cas. Nous désignons par M_P le parent fort de M (qui existe car $M \neq X$).

(1) M et M_P sont premiers. Par conséquent pour tout x de M et pour tout y de $M_P \setminus M$, $M = Z(x, y)$. Comme M est un module fort, il forme à lui tout seul une classe de chevauchement de $\mathcal{Z}(H)$ triviale, il est égal à son support et il est l'unique atome.

(2) M est premier et M_P est complet. Donc pour tout élément x de M et tout élément y de $M_P \setminus M$, M est inclus dans $Z(x, y)$. Remarquons que ces $Z(x, y)$ appartiennent tous à la même composante de chevauchement $\mathcal{C}$ de $\mathcal{Z}(H)$. Et comme M est un module fort de H, $M \subseteq S(\mathcal{C})$ ne peut donc chevaucher aucun membre de $\mathcal{C}$. Qui plus est, pour tout $M \subsetneq N \subseteq S(\mathcal{C})$, N pourrait chevaucher $Z(x, y)$ avec $x \in M$ et $y \in N \setminus M$. En conclusion M est, par définition, un atome de $\mathcal{C}$.

(3) M est complet. Il est aisé de voir que M a k ($\geqslant 3$) enfants forts, $M_1, \ldots, M_k$. Prenons un élément x_i dans chaque M_i. Pour tout i et j nous considérons $Z(x_i, x_j)$. Ces modules ne sont pas tous forts (si tel était le cas M serait premier). Considérons maintenant le graphe de chevauchement de ces modules (les sommets sont les membres de la classe de chevauchement et il y a une arête entre deux sommets si les modules correspondant se chevauchent). Chaque composante connexe de ce graphe correspond à un classe de chevauchement. En référence à la première partie du théorème, le support de chaque classe de chevauchement est un module fort. Si il y a deux classes de chevauchement, le support d'au moins une des deux composantes de chevauchement est un module fort qui est strictement compris entre M et ses enfants M_i dans l'arbre d'inclusion, et comme le graphe de chevauchement a moins une arête, nous aboutissons à une contradiction. Par conséquent il doit y avoir seulement une seule classe de chevauchement, et son support est exactement M.

□

Pour une relation homogène arbitraire, le théorème 2.21 donne les bases d'un algorithme d'énumération de tous les modules forts en temps $O(n^3)$. Cette approche sera présenté en section 4 *p.* 41.

2.3. Relations homogènes particulières.

Nous avons vu dans les sections précédentes ce que nous appelons les relations homogènes arbitraires, et de par leur définition assez large, il est difficile de trouver des propriétés structurelles fortes. C'est pourquoi ici nous considérons des relations homogènes particulières. Ce sont des relations homogènes auxquelles nous imposons quelques restrictions. Ce faisant nous sommes capables de dire plus de choses en terme combinatoire et nous sommes également capables de fournir de meilleurs algorithmes que dans le cas général.

Définition 2.22. Une relation homogène H est :

(1) **Faiblement graphique** si $H(y|xz) \wedge H(z|xy) \Rightarrow H(x|yz)$ pour tout $x, y, z \in X$.

(2) **Faiblement "digraphique"** si $H(s|xy) \wedge H(t|xy) \wedge H(y|sx) \wedge H(y|tx) \Rightarrow H(x|st)$ pour tout $x, y, s, t \quad \in X$.

(3) **Quotient modulaire** si $H(x|st) \Leftrightarrow H(y|st)$ pour tout module M de H, et pour tout $x, y \in M$ et $s, t \notin M$.

PROPOSITION 2.23. *Les relations homogènes faiblement graphiques sont également faiblement digraphiques. De plus il existe des relations homogènes faiblement graphiques qui ne respectent pas la condition du quotient modulaire. Par ailleurs, il existe des relations homogènes qui respectent le quotient modulaire et qui ne sont pas faiblement digraphiques.*

DÉMONSTRATION. Montrons que toutes relations faiblement graphique est également faiblement digraphique. Une relation homogène faiblement graphique vérifie pour tout x, y, z de X : $H(y|xz) \wedge H(z|xy) \Rightarrow H(x|yz)$. Donc pour tout s, t, x, y de X nous avons :
- $H(s|xy) \wedge H(y|sx) \Rightarrow H(x|sy)$ et
- $H(t|xy) \wedge H(y|tx) \Rightarrow H(x|ty)$.

En utilisant la propriété de transitivité de la relation homogène pour x nous en déduisons que $H(x|st)$.

Montrons maintenant qu'il existe des relations homogènes faiblement graphiques qui ne sont pas quotients modulaires. Soit K une relation homogène définie sur $X_k = \{x, y, s, t\}$ avec K :
- $K_x = \{\{y\}, \{s\}, \{t\}\}$,
- $K_y = \{\{x\}, \{s, t\}\}$,
- $K_s = \{\{x, y\}, \{t\}\}$,
- $K_t = \{\{x, y\}, \{s\}\}$.

La relation K est faiblement graphique, (on peut s'en assurer en examinant tous les triplets), mais K n'est pas quotient modulaire en effet on a $\overline{H(x|st)} \wedge H(y|st)$ pour le module $\{x, y\}$.

Montrons maintenant qu'il existe des relations homogènes qui sont quotients modulaires mais qui ne sont pas faiblement digraphique. Pour cela considérons la relation L définie sur l'ensemble $X_L = \{x, y, s, t, z\}$.
- $L_x = \{\{s\}, \{t, y, z\}\}$,
- $L_y = \{\{s, t, x\}, \{z\}\}$,
- $L_s = \{\{x, y\}, \{t, z\}\}$,
- $L_t = \{\{x, y\}, \{s, z\}\}$,
- $L_z = \{\{x\}, \{s, t, y\}\}$.

Comme la relation L est première pour la décomposition modulaire, par vacuité, elle respecte la condition du quotient modulaire. Cependant cette relation n'est pas faiblement digraphique. En effet x, y, s, t constitue un contre-exemple. □

La propriété du quotient modulaire joue un rôle très important dans la partie algorithmique de la décomposition modulaire. En effet, si H est une relation homogène qui vérifie la propriété du quotient modulaire, les éléments dans un module M de H perçoivent de manière uniforme un ensemble A qui n'intersecte pas M : si un élément de M distingue deux éléments de A, alors tous les éléments de M distinguent ces deux éléments de A. Cette propriété associée avec la définition de module nous permet de contracter M en un seul élément, le *quotient* par M, ou tout simplement ne conserver qu'un seul élément de M comme élément représentatif pour le module.

Par conséquent, la récursivité peut être utilisée lorsqu'il s'agit de traiter avec les modules de telles relations.

La propriété du quotient modulaire ainsi que les restrictions (*cf.* définition 2.22) ont été introduite et utilisées pour la décomposition modulaire des graphes par Möhring et Radermacher (1984).

Par la suite nous nous référerons aux relations homogènes équipées de la propriété du quotient modulaire comme étant les *bonnes relations homogènes*, et la section 5 *p.* 52 est consacré à leur étude.

Une autre façon d'imposer des restrictions sur une relation homogène, autre que celle présentée dans la définition 2.22, est de restreindre le nombre de classes intervenant dans la relation homogène. Nous appelons cela la congruence locale d'une relation homogène. Une définition plus formelle est présentée en définition 2.4.

Les relations homogènes de congruence locale 2 jouent un rôle particulier en théorie des graphes car elles contiennent à la fois les relations homogènes standard des graphes non-orienté et des tournois (*cf.* la section suivante). De plus ces relations ont la bonne propriété suivante :

PROPOSITION 2.24. *Toutes les relations homogènes H faiblement graphiques et de congruence locale égale à 2 sont quotients modulaires.*

DÉMONSTRATION. Procédons par l'absurde et supposons que nous avons une relation homogène de congruence locale égale à 2 qui soit faiblement graphique, mais qui ne respecte pas la condition du quotient modulaire. Soit H une relation homogène définie sur $X = \{x, y, s, t\}$, tel que $\{x, y\}$ soit un module et que nous ayons $H(x|st) \wedge \overline{H(y|st)}$. Montrons que nous avons à la fois $\overline{H(y|xs)}$ et $\overline{H(y|xt)}$.

Supposons sans perte de généralité que nous ayons $H(y|xs)$. Alors la transitivité de la relation H_y implique que nous avons $\overline{H(y|xt)}$ (car nous avons par hypothèse $\overline{H(y|st)}$).

De plus comme $\{x, y\}$ est un module de H nous avons $H(s|xy)$. La propriété faiblement graphique implique $H(x|sy)$ et la transitivité de la relation H_x nous donne $H(x|ty)$. Mais nous devrions donc avoir $H(x|ty)$, $H(t|xy)$ (comme $\{x, y\}$ est un module de H) et $\overline{H(y|xt)}$ ce qui contredit la propriété faiblement graphique. Par conséquent nous avons $\overline{H(y|st)}$, $\overline{H(y|xs)}$ et $\overline{H(y|xt)}$ finalement la congruence locale de y est de 3. □

2.4. Relations homogènes standard.

Définition 2.25 (Relation Homogène Standard). La relation homogène standard $H(G)$ d'un graphe dirigé $G = (X, A)$ est définie telle que pour tout x, u, v éléments de X, $H(G)(x|uv)$ est vrai si et seulement si les deux conditions suivantes sont satisfaites :

– Soit à la fois u et v sont des voisins entrants de x soit aucun.
– Soit à la fois u et v sont des voisins sortants de x soit aucun.

Étant donné un graphe dirigé, ou plus généralement une 2-structure, la relation standard associée est définie dans la définition 2.25. Ces relations sont particulières et satisfont les propriétés (fondamentales) suivantes :

Proposition 2.26. *La relation standard des 2-structure est quotient modulaire. En particulier ce résultat est vrai pour les graphes, les tournois, les graphes orientés ou dirigés.*

La proposition précédente a d'importantes conséquences d'un point de vue algorithmique qui seront détaillées en section 5.

En ce qui concerne les appellations des relations homogènes *"faiblement graphiques"* et *"faiblement digraphiques"*, elles sont motivées par la proposition suivante. Une 2-structure symétrique est une clique (graphe non-orienté) où toutes les arêtes sont colorés (*cf.* définition 1.27 *p.* 15 pour plus de détails).

Proposition 2.27. *La relation homogène standard d'un graphe dirigé, resp. une 2-structure est faiblement digraphique. La relation homogène standard d'un graphe non-orienté, resp. une 2-structure symétrique est faiblement graphique.*

Nous considérons maintenant la question inverse, à savoir : étant donné une relation homogène H définie sur l'ensemble fini X, existe-t-il un graphe non-orienté ou un tournois qui admet H comme relation homogène standard ? Cette question semble assez naturelle et se rapproche de ce qui est fait pour les matroïdes à savoir étant donné un matroïde, savoir si le matroïde est graphique, et éventuellement donner un graphe qui réalise ce matroïde. Ce problème assez naturel trouve ses racines au commencement même de la théorie des matroïdes. Le premier à apporter une réponse à cette question fut Whitney (1933). Ce n'est que récemment que Truemper (1980) donna une preuve simple de ce résultat. Elles est d'ailleurs reprise dans le livre de Oxley (1992).

Nous remarquerons que dans le cas des graphes non orientés, si H est une relation graphique il existe au moins deux graphes qui réalisent cette relation en effet : si on note $H(G)$ la relation homogène H obtenue à partir de G on peut facilement se convaincre que $H(G) \cong H(\overline{G})$. De plus on peut montrer que seulement ces deux graphes réalisent la relation homogène.

DÉFINITION 2.28 (Relation Homogène *Graphique*). Une relation homogène H définie sur l'ensemble fini X est graphique si :

- H est de congruence locale égale à 2.
- et pour tout triplet a, b, c de X, $H[\{a, b, c\}]$ a exactement 0 ou 2 éléments de congruence locale égale à 2.

DÉFINITION 2.29 (Relation Homogène *"Tournamentale"*). Une relation homogène H définie sur l'ensemble fini X est tournamentale si :

- H est de congruence locale égale à 2.
- et pour tout triplet a, b, c de X, $H[\{a, b, c\}]$ a exactement 1 ou 3 éléments de congruence locale égale à 2.

THÉORÈME 2.30. *H est la relation homogène standard d'un graphe non orienté si et seulement si H est graphique.*

DÉMONSTRATION.

Il est facile de voir que si $G = (X, E)$ est un graphe non orienté, alors $H(G)$ est graphique. Considérons maintenant la réciproque : étant donné une relation homogène graphique H définie sur X, montrons que s'il existe au moins un graphe non orienté $G = (X, E)$ tel que $H = H(G)$. Pour cela nous considérons un élément x de X. Et soit C_x une des deux classes d'équivalence de H_x (il en existe toujours au moins une). On définit la matrice carrée M de taille $n \times n$ de la manière suivante :

$M[x, y] = 1$ si y appartient à C_x et 0 sinon.

Ensuite pour tout x' différent de x on définit $M[x', y] = 1$ si y appartient à $C_{x'}$ et 0 sinon, où $C_{x'}$ est la classe de $H_{x'}$ qui contient x.

Supposons que M ne soit pas symétrique, cela signifie qu'existent deux éléments distincts y et z tous deux différents de x tel que $M[y, z] = 1$ et $M[z, y] = 0$. Mais alors la relation homogène $H[\{x, y, z\}]$ aurait 1 ou 3 éléments de congruence locale égale à 2. Ce qui n'est pas possible.

Par conséquent M est une matrice $\{0, 1\}$ symétrique qui peut être vue comme la matrice d'adjacence d'un graphe non orienté. Il est alors direct de voir que H est la relation homogène standard de G. □

COROLLAIRE 2.31 (Motifs interdits pour les graphes non-orientés). *Du théorème précédent, nous déduisons qu'une relation est graphique si et seulement si elle ne contient pas les deux motifs suivants comme relation homogènes induites :*

(1) $H(x|yz) \wedge H(y|xz) \wedge \overline{H(z|xy)}$,
(2) $\overline{H(x|yz)} \wedge \overline{H(y|xz)} \wedge \overline{H(z|xy)}$.

THÉORÈME 2.32. *H est la relation homogène standard d'un tournois si et seulement si H est tournamentale.*

DÉMONSTRATION. La preuve est similaire à celle du théorème 2.30, on construit la matrice $\{1, -1\}$ et en utilisant la propriété que la matrice d'adjacence d'un tournoi est antisymétrique. □

COROLLAIRE 2.33 (Motifs interdits pour les tournois). *Du théorème précédent, nous déduisons que H est une relation homogène tournamentale si et seulement si elle contient pas les deux motifs suivants comme relation homogène induite.*

(1) $\overline{H(x|yz)} \wedge \overline{H(y|xz)} \wedge H(z|xy)$
(2) $H(x|yz) \wedge H(y|xz) \wedge H(z|xy)$

COROLLAIRE 2.34. *Étant donné une relation homogène H définie sur X, on peut tester en temps $O(|X^3|)$ si la relation homogène H admet un graphe G ou un tournois T tel que $H(G) = H$ ou $H(T) = H$.*

DÉMONSTRATION. On vérifie d'abord que tous les éléments sont de congruence locale 2. Il suffit ensuite de vérifier pour tous les triplets la propriété correspondante de la relation induite par chaque triplet. □

REMARQUE 2.35. Nous pouvons améliorer le résultat précédent en $O(n^2)$ et construire le graphe (*resp.* tournois) correspondant à la relation homogène ou le cas échéant exhiber un motif interdit pour la propriété concernée. Si on considère le cas des graphes non orienté, on choisit un élément x, et $H_x = \{H_x^1, H_x^2\}$ définit le voisinage et le non voisinage de x, sans perte de généralité pour x on choisit que H_x^1 représente le voisinage de x. par conséquent pour tous les voisins y de x, la classe H_y^i (pour $i \in \{1, 2\}$) est forcé. En clair le voisinage de y est la classe H_y^i tel que $x \in H_y^i$. En propageant de proche en proche ces contraintes, on construit progressivement le graphe. Soit nous parvenons à construire le graphe sans violer les contraintes d'appartenance de classes, soit nous détectons une contradiction.

REMARQUE 2.36. Il existe des relations homogènes de congruence locale égale à 2 qui ne sont ni graphiques ni tournamentales.

Considérons la relation homogène suivante : Soit $X = \{a, b, c, d\}$ et H défini comme suit :

– $H_a = \{\{b, c, d\}\}$,
– $H_b = \{\{a, c\}, \{d\}\}$,
– $H_c = \{\{a, d\}, \{b\}\}$,
– $H_d = \{\{a, c\}, \{b\}\}$.

Nous remarquons que $H[\{a, b, c\}]$ n'est pas graphique, donc H n'est pas graphique et que $H[\{a, c, d\}]$ n'est pas tournamentale donc H n'est ni graphique ni tournamentale et pourtant de congruence locale 2.

Notons que si une relation homogène graphique (*resp.* tournamentale) nous est donnée comme n ensembles de classes d'équivalence de H_x, alors nous pouvons construire un graphe (*resp.* tournoi) sous forme de liste d'adjacence en temps $O(n^2)$.

Pour les graphes, on choisit quelle est la classe du premier élément x qui représente le voisinage. Pour tout autre sommet u différent de x la classe qui contient v va être son voisinage si u est un voisin de v, et son non-voisinage sinon. Ensuite il suffit de supprimer pour chaque élément la classe qui définit le non-voisinage. Cela se fait en temps $O(n)$ et comme nous procédons de la sorte pour tous les éléments nous obtenons une complexité globale en temps $O(n^2)$.

Nous pouvons procéder de manière similaire pour traiter les relations homogènes tournamentales.

REMARQUE 2.37. Toute relation homogène H de congruence locale 2, est la relation homogène d'un graphe dirigé.

DÉMONSTRATION. Pour chaque élément x de H on choisit une classe C_x de H_x nous construisons la matrice carrée $M_{n,n}$ avec pour tout $M[x, y] = 1$ si y appartient à C_x, 0 sinon. Nous pouvons construire le graphe dirigé $G = (X, E)$ de la manière suivante pour tout x, y tel que $M[x, y] = 1$, $E \leftarrow E \cup (xy)$. Autrement dit nous attachons un arc sortant de x vers y. □

Nous avons montré que l'on pouvait complètement caractériser et reconnaître les relations homogènes graphiques et tournamentales, cependant la caractérisation des graphes orientés et dirigés ou plus généralement des 2-structure reste à faire. Un récapitulatif des différentes familles de relations homogènes est donné en figure 2.3.

3. Partitivité et théorème de décomposition

Une généralisation de la décomposition modulaire des graphes, introduite par Chein et al. (1981), bien que moins générale que la décomposition modulaire des relations homogènes, sont les familles partitives. Une présentation formelle est présentée dans la définition 1.18.

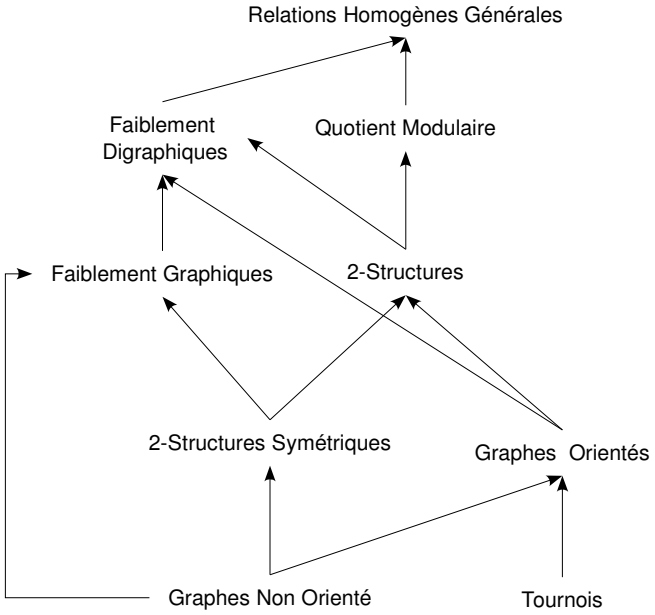


Figure 2.3. Les relations d'inclusions entre les différents types de relations homogènes

Rappelons simplement qu'une famille $\mathscr{F}$ de sous-ensembles d'un ensemble fini X est partitive si elle comporte tous les singletons $\{x\}$ et l'ensemble support X. De plus elle est close par l'union, l'intersection, la différence et la différence symétrique d'éléments qui se chevauchent. Nous distinguons les familles faiblement partitives, lesquelles ne comportent plus la différence symétrique.

Soit $\mathscr{F}$ une famille faiblement partitive définie sur X. Comme nous l'avons mentionné plus tôt, les éléments forts de la famille peuvent être ordonnés par inclusion dans un arbre, celui que nous appelons l'arbre de décomposition généralisé (*cf.* Section. 2.2). Dans cet arbre, les enfants, avec la notion naturelle de parenté dans les arbres, d'un noeud interne M sont par définition des éléments forts de $\mathscr{F}$, et sont également des enfants forts de l'élément fort M de $\mathscr{F}$ dans le sens de la section 2.2.

Nous définissons trois types d'éléments forts de $\mathscr{F}$, à savoir trois types de noeuds (internes) de l'arbre.

- Noeuds *Premiers* qui n'ont pas d'enfant faible.
- Noeuds *Complets* : toute union d'enfants forts du noeud appartiennent à $\mathscr{F}$.
- Noeuds *Linéaires*, il y a un ordre (total *i.e.* une permutation) des enfants forts du noeud tel que l'union de ces fils appartient à la famille si et seulement si il constituent un intervalle dans la permutation.

THÉORÈME 2.38 (Chein et al. (1981)). *Dans une famille partitive, il y a seulement des noeuds premiers et des noeuds complets. Dans une famille faiblement partitive, il y a des noeuds premiers, complets et linéaires.*

En conséquence de quoi l'arbre généralisé de décomposition admet un codage qui occupe un espace $O(n)$. Il suffit de typer les noeuds (internes) soit en noeud complet, en noeud linéaire, ou en noeud premier. Nous appelons alors cet arbre, l'arbre de décomposition de la famille.

À partir de cet arbre il est possible de renvoyer la liste de tous les éléments faibles de la famille $\mathscr{F}$ en procédant à de simples combinaisons des noeuds complets et linéaires.

PROPOSITION 2.39. *Les modules d'une relation homogène faiblement graphique (resp. faiblement digraphique) forment un famille partitive (resp. faiblement partitive).*

DÉMONSTRATION. Le lemme 2.12 fournit la clôture par union et par intersection d'éléments qui se chevauchent. Il faut maintenant montrer la différence et la différence symétrique.

Dans un premier temps considérons les relations homogènes faiblement digraphiques.

Soient $A \in \mathscr{M}_H$ et $B \in \mathscr{M}_H$ deux modules de H qui se chevauchent. Supposons qu'il existe un casseur s de $A \setminus B$, il existe x et y de $A \setminus B$ tel que $\overline{H(s|xy)}$. De plus s appartient à $A \cap B$, dans le cas contraire ce serait un casseur de A en entier.

Finalement comme $A \mathbin{@} B$, il existe un élément t appartenant à $B \setminus A$. Nous avons $H(x|st)$, $H(y|st)$ et $H(t|sy)$ et $H(t|xy)$. En clair H n'est pas faiblement digraphique. Par conséquent la famille des modules $\mathscr{M}_H$ d'une relation faiblement digraphique est faiblement partitive.

En ce qui concerne les relations homogènes faiblement graphiques. Considérons maintenant que z est un casseur de $A \Delta B$. Par conséquent z appartient toujours à $A \cap B$, et il existe un élément $x \in A$ et $y \in B$ $\overline{H(z|xy)}$, $H(x|yz)$ et $H(y|xz)$, par conséquent H n'est pas faiblement graphique. Par suite de quoi la famille des modules d'une relation homogène faiblement graphique est partitive. □

Cela a pour conséquence que la famille des modules d'une relation homogène standard forme une famille faiblement partitive, car une telle relation est toujours faiblement digraphique (*cf.* section 2.4). Plus généralement, nous montrerons par la suite que la famille des modules d'une relation homogène, qui satisfait la propriété du quotient modulaire (*cf.* section 2.3), forme un famille faiblement partitive.

Rappelons également qu'une famille faiblement digraphique ne respecte pas nécessairement la propriété du quotient modulaire (*cf.* proposition 2.23).

4. Algorithmes pour les relations homogènes arbitraires

Nous nous intéressons au problème suivant, étant donné une relation homogène H définie sur un ensemble X, construire les outils algorithmiques nécessaires pour construire l'arbre de décomposition modulaire généralisé.

Nous présentons un algorithme pour calculer cet arbre, sa complexité est en $O(n^3)$, l'algorithme est présenté section 4.5

4.1. Structures de données.

Pour représenter en mémoire une relation homogène, et selon la définition 2.1, deux solutions s'offrent à nous :

La première consiste à considérer la relation homogène sous la forme d'une matrice carrée $A_{n,n}$ avec des valeurs dans $\{1, \ldots, n\}$.

L'ensemble support X est de la forme $\{x_1, \ldots, x_n\}$. Chaque classe d'équivalence de la relation H_{x_i} va se voir assigné un nombre de de 1 à n. Alors $A[i, j]$ a la valeur k si et seulement si x_j appartient à la classe k de H_{x_i}. Cette représentation permet de tester en temps constant si $H(x_i|x_p x_q)$ en vérifiant que $A[i, p] = A[i, q]$. Cette représentation a l'inconvénient d'être très coûteuse en temps lorsque l'on souhaite récupérer l'intégralité d'une classe d'équivalence donnée (*i.e.* donner tous les éléments d'une même classe de H_{x_i}) peut prendre dans le pire des cas un temps $O(n)$.

La seconde solution permet de pallier à cet inconvénient en utilisant une représentation à base de listes. Pour chaque élément x de X, nous associons une liste de classes d'équivalence de la relation H_x.

Cette liste peut se permettre d'ignorer une classe C_x, en effet comme H_x définit une partition de $X \setminus \{x\}$, on peut aisément en supprimer une et la reconstruire à partir des autres. Nous pourrons par exemple ignorer la plus grande classe.

Par conséquent l'espace total utilisé est en taille $O(n + m)$, avec $n = |X|$ et $m = \sum_{x \in X} n - |C_x|$. Cette représentation permet d'accéder en temps constant à une classe d'équivalence de H_x, par contre tester si deux éléments y et z sont homogènes vis à vis de x se fait, dans le pire des cas, en temps $O(n - |C_x|)$.

Nous remarquerons que pour une relation homogène, il est direct de construire en temps $O(n^2)$ la représentation en liste, étant donné une représentation sous forme de matrice et réciproquement.

Nota Bene : Dans la suite du chapitre nous n'utiliserons que la représentation en matrice comme entrée.

Un exemple de représentation d'une relation homogène est présentée au début de ce chapitre en figure 2.1 *p.* 25.

4.2. Plus petit module contenu dans un sous-ensemble.

Soit S un sous-ensemble non vide de X. Comme $\mathscr{M}_H$ est close par intersection (d'éléments chevauchant), il existe un unique plus petit module contenant S, à savoir l'intersection de tous les modules qui contiennent S, noté alors $SM(S)$.

Entrées : $S \subseteq V$
Sorties : $SM(S)$, le plus petit module contenant S
1 **début**
2 Soit x un élément de S
3 $M \leftarrow \{x\}$
4 $F \leftarrow S \setminus \{x\}$
5 **tant que** F *non vide* **faire**
6 prendre un élément y in F
7 $F \leftarrow F \setminus \{y\}$
8 $M \leftarrow M \cup \{y\}$
9 **pour chaque** *élément* $z \notin (M \cup F)$ **faire**
10 **si** $\overline{H(z|xy)}$ **alors**
11 $F := F \cup \{z\}$
12 Renvoyer M (qui vaut maintenant $SM(S)$)
13 **fin**

Algorithme 1 : Plus petit module contenant S

THÉORÈME 2.40. *L'algorithme 1 calcule $SM(S)$ en temps $O(|X|.|SM(S)|) = O(|X|^2)$.*

DÉMONSTRATION.
Complexité : La boucle **tant que** ligne 5 s'exécute $|M|-1$ fois et la boucle **pour chaque** ligne 9 s'exécute n fois. Par conséquent l'algorithme se termine en un temps $O(n^2)$.
Correction : L'algorithme maintient l'invariant suivant, tous les casseurs de M sont dans F.

Quand M est remplacé par $M \cup \{y\}$, en s'appuyant sur la propriété de transitivité de H_x, tout casseur de $M \cup \{y\}$ soit distingue x de y, soit il est déjà dans F. Par conséquent l'algorithme termine sur un module qui contient S. En effet l'algorithme s'arrête quand F est vide, donc il n'y a plus de casseur, et comme initialement $F \cup M = S$. Nous avons donc $S \subseteq M \subseteq SM(S)$.

Maintenant si M est différent de $SM(S)$, soit s le premier élément de $M \setminus SM(S)$ ajouté à F (finalement ajouté à M). Il distinguait deux élément x et y de $SM(S)$, contredisant son homogénéité. Par conséquent $SM(S) = M$. □

PROPOSITION 2.41. *Soit x un élément de X. Et comme $\mathscr{M}_H$ est fermé par l'union d'éléments chevauchants, il existe une unique partition de $X \setminus \{x\}$ en $\{S_1, \ldots, S_k\}$ telle que chaque S_i est un module de H et est maximal (vis à vis de l'inclusion) dans $\mathscr{M}_H$.*

DÉMONSTRATION. Il est clair que tout module ne contenant pas x ne peut chevaucher ou contenir plus d'une classe de H_x, sinon x constituerait un casseur pour cet ensemble.

Montrons maintenant que cette partition est unique. Procédons par l'absurde, et supposons qu'il existe deux partitions différentes $\mathcal{P}_1$ et $\mathcal{P}_2$, et supposons qu'elles soient maximales vis à vis de l'inclusion. Comme $\mathscr{M}_H$ est fermé par union d'éléments chevauchants, nous pouvons construire $\mathcal{P}$ à partir de $\mathcal{P}_1$ et $\mathcal{P}_2$ en plaçant dans $\mathcal{P}$ l'union de deux éléments qui se chevauchent dans les deux partitions précédentes. Nous obtenons donc une partition $\mathcal{P}$ qui contient strictement les deux autres ce qui contredit l'hypothèse de maximalité. □

Nous appelons $MaxM(x)$ cette partition des modules maximaux ne contenant pas x. Nous présentons ci-après un algorithme de partitionnement pour la calculer.

L'affinage de partition est un concept très puissant en algorithmique et de nombreux algorithmes utilisent ce concept, on pourra citer le fameux algorithme de minimisation d'automate déterministe dû à Hopcroft, ainsi que le célèbre algorithme du **Quick Sort**, mais ce concept a vraiment été reconnu dans le papier de Paige et Tarjan (1987) et Paul (1998).

Une illustration d'affinage de partition est donné en figure 2.4.

LEMME 2.42. *Tout module ne contenant pas x (en particulier les maximaux) est inclus dans une des classes d'équivalence de H_x.*

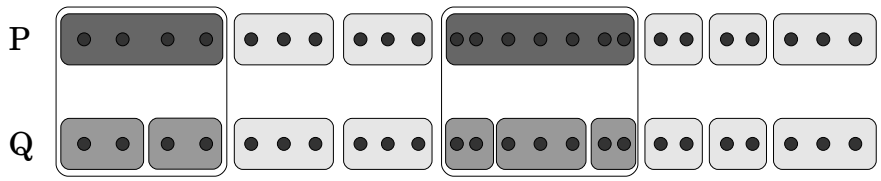


Figure 2.4. Affinage de partition : La partition Q est plus fine que la partition P

Par conséquent l'algorithme va commencer avec la partition initiale $\mathcal{P} = \{H_x^1, \ldots, H_x^k\}$ des classes d'équivalence de H_x. Ensuite la partition est affinée (des classes sont partitionnées en morceaux plus petits) en utilisant la règle qui suit. Soit y un élément, appelé le pivot, et Y la partie de $\mathcal{P}$ qui contient y.

Règle 1 : Couper chaque partie A de $\mathcal{P}$ distincte de Y en $\{A \cap H_y^1, \ldots, A \cap H_y^k\}$.

Nous remarquerons qu'une partie est coupée si et seulement si y constitue un casseur pour cette partie.

LEMME 2.43. *En commençant avec la partion $\mathcal{P}_0 = \{H_x^1, \ldots, H_x^k\}$, l'application de la règle 1 (pour n'importe quel pivot et quel que soit l'ordre de choix) jusqu'à ce que plus aucune des parties ne soient coupées (i.e. un point fixe), produit la partition $MaxM(x)$.*

DÉMONSTRATION. L'affinage de partition se termine quand plus aucun pivot ne permet de couper quelque partie que ce soit, ou plus plus précisément quand chaque partie est un module de H.

Supposons maintenant qu'il existe un module M, obtenu par cet algorithme, qui ne soit pas maximal vis à vis de l'inclusion : en d'autre terme il existe un module M' de H qui contient strictement M. Et M' est lui même contenu dans une classe d'équivalence de H_x (*cf.* lemme 2.42).

Considérons maintenant le premier pivot y qui a cassé M' en plusieurs parties. Cet élément ne peut pas être à l'extérieur de M' car M' est un module de H, de plus y ne peut pas être à l'intérieur de M' car la règle 1 interdit de casser sa propre partie. Donc M' a tout de même été cassé, d'où une contradiction. □

Nous avons montré que l'application répétée de la règle 1 permet de calculer $MaxM(x)$. Nous allons maintenant montrer comment implémenter cela de manière efficace dans l'algorithme suivant.

Par convention $\mathcal{P}_i$ représentera la partition obtenue après la i-ème application de la règle 1. y sera un élément pivot donné, et Y_i la partie de $\mathcal{P}_i$ qui contient y.

Nous dirons qu'une partie B de $\mathcal{P}_j$ descend d'une partie A de $\mathcal{P}_i$ si $i < j$ et $B \subset A$.

Il est clair qu'une fois qu'un élément y a été choisi comme pivot à l'étape i, y ne distingue plus aucune partie de $\mathcal{P}_i$ hormis Y_i. Si y est choisi comme pivot à l'étape $j > i$, y peut seulement couper les parties de $\mathcal{P}_{j-1}$ qui descendent de Y_i. Seules ces parties n'ont pas été considérées de l'application la règle 1. Mais Y_j lui même n'a pas à être traité.

Supposons maintenant que pour une partie A, nous puissions couper celle-ci en temps $O(|A|)$ en appliquant la règle 1. Le temps nécessaire pour appliquer la règle 1 à l'étape j serait proportionnel à $O(|Y_i - Y_j|)$, la somme des tailles des parties qui descendent de Y_i moins Y_j.

Le temps nécessaire pour toutes les cassures (les affinages) de chaque partie avec y comme pivot prend un temps $O(n)$, ce qui entraîne une complexité globale en $O(n^2)$. Cette procédure est implémentée dans l'algorithme 2.

Nous allons maintenant implémenter de manière efficace le lemme précédent. Mais pour cela nous avons besoin de quelques notations et définitions supplémentaires.

Soit $\mathcal{P}_i$ la partition obtenue à partir de la partition initiale après la i-ème itération de la **règle 1**. Nous disons qu'une partie B de la partition $\mathcal{P}_j$ descend d'une partie A de la partition $\mathcal{P}_i$ si $i < j$ et $B \subset A$.

Il est facile de voir en observant la **règle 1**, que lorsque un élément y a été choisi comme pivot à l'étape i, et qu'à une étape ultérieure j $(i < j)$ y est choisi une nouvelle fois, les parties qui peuvent être coupées par y sont les classes de $\mathcal{P}_{j-1}$ qui descendent de Y_i. En effet comme la règle 1 ne coupe pas sa propre classe, seuls les descendants de Y_i peuvent

Entrées : $\mathcal{P}_0 = \{H_x^1, \ldots, H_x^k\}$
Sorties : $\mathcal{P}_i = MaxM(x)$
début
  pour chaque *groupe* G **faire**
    pour chaque *partie* C *de* G **faire**
      Calculer l'ensemble Z des éléments dans G qui ne sont pas dans C
      pour chaque *élément* y *de* C **faire**
        Partitionner Z selon la classe d'équivalence de H_y
        Ajouter chaque partition à la liste des ensembles à affiner
  Changer les limites des groupes au limites des parties (de $\mathcal{P}_{i-1}$ à $\mathcal{P}_i$)
  for *ensemble d'affinage* R *de la liste* **do**
    Supprimer R de la liste
    Affiner $\mathcal{P}_i$ en utilisant R
fin

Algorithme 2 : $MaxM$: Modules maximaux ne contenant pas x

être distingués par y. Par conséquent seuls les descendants de Y_i doivent être examinés, excepté bien sûr Y_j.

Supposons maintenant que nous soyons capables de traiter (*i.e.* appliquer la **règle 1** sur cette partie) une partie A en temps $O(|A|)$. Par conséquent, à chaque étape j, le temps de traitement nécessaire est $O(|Y_i| - |Y_j|)$, la somme des tailles des parties qui descendent de Y_i excepté Y_j. Le temps de tous les affinages avec y comme pivot est $O(n)$.

Ce qui nous donne un algorithme global ayant pour complexité $O(n^2)$. Il s'agit de l'algorithme 2.

Nous présentons maintenant les détails de l'implémentation, et quelles sont les complexités des différentes sous-routines.

Nous considérons maintenant que notre partition est codée sous forme de listes chaînées. À chaque étape les nouvelles parties créées à l'issue d'une application de la **règle 1** remplacent l'ancienne partie. Et se suivent consécutivement dans la liste.

Par conséquent pour chaque élément pivot y il suffit de stocker seulement deux pointeurs pour indiquer quelle sont les parties qui descendent de Y_i. Un premier pointeur sur la première partie qui descend de Y_i et un autre pointeur sur la dernière partie. Il suffit ensuite, de traiter toutes les parties (sauf Y_j) qui se situent entre ces deux pointeurs (*cf.* figure 2.5).

Nous allons maintenant expliquer comment nous pouvons effectivement traiter une partie A en un temps $O(|A|)$. Il s'agit en réalité d'une technique classique d'affinage de partition.

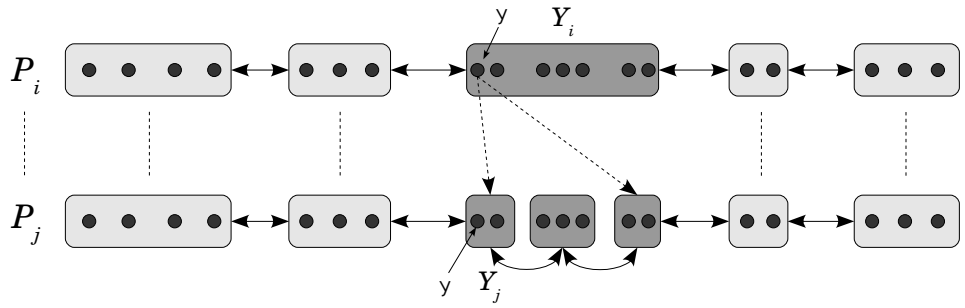


Figure 2.5. Affinage de partition et gestion des pointeurs pour le pivot.

Si le pivot considéré est l'élément y et que les classes d'équivalence de H_y sont numérotées de 1 à k. Il est possible de trier A en temps $O(|A| + k)$ avec un tri par "bucket", et chaque "bucket" donne lieu à une nouvelle partie qui descend de A. Si $|A| < k$, il est nécessaire de renuméroter les classes d'équivalence de H_y de 1 à k' où $k' \leqslant |A|$ avant de faire le tri par "bucket". Un premier passage sur A marque les classes d'équivalences utilisées, (leurs numéros). Un second passage démarque un numéro utilisé la première fois et le remplace par un nouveau numéro (qui est incrémenté au fur et à mesure) qui est plus petit que $|A|$. Le vecteur des classes d'équivalences est initialisé une seule fois en temps $O(k)$.

Le dernier point à considérer est l'ordre dans lequel les pivots sont sélectionnés. Utiliser tous les éléments une fois comme pivot répéter cela n fois est suffisant. Ce qui fait en tout $O(n^2)$ usage des pivots. Une façon plus subtile consiste à considérer y seulement si Y_i a été coupé en conservant une liste des pivots actifs.

Nous définissons maintenant une mesure qui nous sera utile par la suite pour mesurer la complexité.

Définition 2.44. Soit $\mathcal{P}$ une partition de X. $Q(\mathcal{P})$ est le nombre de paires $\{x, y\}$ telles que x et y ne sont pas dans la même partie (ou classe) de $\mathcal{P}$.

$Q(\mathcal{P})$ varie entre 1 (quand $\mathcal{P} = \{X\}$) et $\dfrac{n(n-1)}{2}$ (quand $\mathcal{P}$ est la partition en singletons).

Théorème 2.45. *$MaxM(x)$ peut être calculé en temps $\Theta(Q(MaxM(x))) = O(n^2)$.*

Démonstration. En ce qui concerne la correction de l'algorithme il suffit de s'assurer que l'algorithme 2 implémente correctement le lemme 2.43. En ce qui concerne la complexité, remarquons que pour chaque pivot y, un élément z est placé dans Z seulement une fois. De plus il est placé dans Z seulement si y et z ne sont pas dans une même partie. Par conséquent, à chaque étape, nous affinons Z selon les classes d'équivalences H_y de y,

nous utilisons donc tous les ensembles générés par y ce qui prend un temps $O(|Z|)$. En conséquence de quoi l'algorithme s'exécute en temps $O(Q(MaxM(x)))$. □

4.3. Test de primalité.

Nous présentons maintenant un algorithme qui teste si la relation homogène considérée est première, à savoir si les seuls modules de la relation sont l'ensemble X et tous les singletons.

THÉORÈME 2.46. *Savoir si une relation homogène H est première peut être obtenue en temps $O(n^2)$.*

DÉMONSTRATION. Si $|X| < 2$ la réponse est clairement oui. Dans le cas contraire, considérons deux éléments x et y de X. Nous pouvons, en un temps $O(n^2)$, utiliser l'algorithme 2 pour calculer la partition des modules maximaux qui ne contiennent pas x. Si un des modules ainsi obtenu est non trivial, on en déduit que H n'est pas première.

Dans le cas contraire tous les modules non triviaux de H vont contenir x. On calcule maintenant, avec le même algorithme, la partition des modules maximaux qui ne contiennent pas y, on procède de manière analogue, si il existe dans $\mathcal{P}_y$ des modules non triviaux, alors H n'est pas première. Sinon cela signifie que tous les modules non triviaux (s'ils existent) contiennent y et x. Par conséquent on peut utiliser l'algorithme 1. Nous calculons le plus petit module qui contient à la fois x et y ($SM(\{x, y\})$). La relation homogène H est première si et seulement si $SM(\{x, y\}) = X$, à savoir le plus petit module qui contient x et y est le module trivial $\{X\}$. □

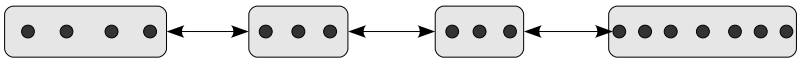


Figure 2.6. Représentation d'une partition sous forme de liste chaînées.

4.4. Énumération des modules forts.

Le théorème 2.21 conduit naturellement à un algorithme :

THÉORÈME 2.47. *Les modules forts d'une relation homogène H définie sur l'ensemble X peuvent être énumérés en temps $O(n^3)$.*

DÉMONSTRATION. Dans un premier temps nous calculons les ensembles $MaxM(x)$ pour tous les éléments x de X. Nous réunissons tous ces ensembles pour former la famille $\mathcal{Z}(H)$(telle que définie pour le théorème 2.21). Cela peut être fait en temps $O(n^3)$ en utilisant l'algorithme 2, n fois.

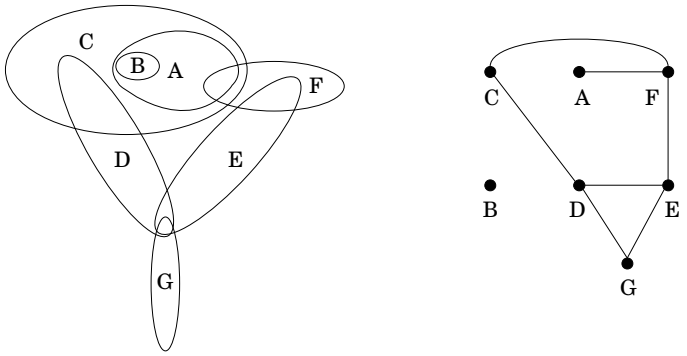


Figure 2.7. Familles d'ensembles et son graphe de chevauchement, A et C ne sont pas reliés dans le graphe mais appartiennent à la même composante.

La taille de cette famille (*i.e.* la somme des cardinalités de chacun des sous-ensembles) est en $O(n^2)$ car ils forment n partitions de taille n. Nous calculons ensuite les composantes de chevauchement en utilisant, au choix, l'algorithme de Dahlhaus (2000) ou bien Charbit et al. (2008) (*cf.* Chapitre. 4 *p.* 99). Les composantes de chevauchement peuvent être trouvées en temps linéaire en la taille de la famille, à savoir $O(n^2)$. Selon la proposition 2.20 il y a au plus n classes de chevauchements non triviales.

Pour chacune de ces classes, il est facile de calculer leurs supports, et en un temps $O(n^2)$ de calculer tous ses atomes. Pour ce faire, nous considérons le vecteur caractéristique de chaque élément x vis à vis des classes de chevauchement. Les atomes sont les éléments avec le même vecteur caractéristique.

Par la suite les $O(n^2)$ supports et atomes doivent être triés par ordre d'inclusion dans l'arbre des modules forts. Cela peut être fait en temps $O(n^3)$ en utilisant la même technique de tri.

Finalement, les mauvais atomes - ceux qui ne sont pas des modules forts - doivent être supprimés de l'arbre. Selon le premier point du théorème 2.21 les atomes qui sont des modules sont des modules forts. Il nous reste seulement à procéder à $O(n)$ tests sur tous les noeuds de l'arbre et de tester pour chacun, lesquels sont des modules et lesquels ne le sont pas. Ce qui peut être calculé en temps $O(n^2)$ pour chacun. □

4.5. Calcul de l'arbre de décomposition généralisé.

Nous présentons maintenant une manière de calculer l'arbre de décomposition généralisé pourvu qu'une permutation factorisante nous soit donnée.

La notion de permutation factorisante dans le cas des graphes Capelle (1997) fût introduite pour proposer une alternative au calcul de l'arbre de décomposition modulaire

sans avoir, au préalable, à calculer les modules maximaux ne contenant pas un sommet x, Bui-Xuan et al. (2005) ; Capelle et al. (2002) ; McConnell et Montgolfier (2005a) ; Möhring (1985). Nous montrons ici comment cette notion peut être étendue aux relations homogènes.

DÉFINITION 2.48 (Permutation Factorisante). Une permutation factorisante d'une relation homogène définie sur l'ensemble X, est une permutation σ des éléments de X, telle que pour tout module fort M de H, M est un facteur de σ (*i.e.* des éléments consécutifs de la permutation).

Ici nous considérons le problème qui consiste, étant donné une relation homogène H définie sur X et σ une permutation factorisante, à calculer l'arbre de décomposition généralisé de H. Il est clair que ce problème est réglé par le théorème 2.47. Cependant nous présentons maintenant une solution plus efficace. La complexité de l'algorithme ainsi obtenu est temps $O(n^2)$. Cette approche est proche de l'algorithme présenté par Uno et Yagiura (2000).

En réalité, l'appellation de permutation factorisante vient de la caractérisation suivante. Sans perte de généralité, nous noterons les éléments de X par $X = \{1, 2, \ldots, n\}$.

PROPOSITION 2.49. *Si σ est une permutation factorisante d'une relation homogène H définie sur l'ensemble fini X, alors tout module fort M de H est un intervalle de σ, à savoir M est de la forme $\{\sigma_i, \sigma_{i+1}, \ldots, \sigma_j\}$*

En d'autres termes, pour énumérer tous les modules forts de H, il suffit de trouver parmi les intervalles de σ ceux qui sont des modules forts.

I_{ij} représente l'intervalle de la permutation σ, $I_{ij} = \{\sigma_i, \sigma_{i+1}, \ldots, \sigma_j\}$, et nous noterons $\mathcal{S}_{ij}$ l' casseur de I_{ij}.

PROPOSITION 2.50. $\mathcal{S}_{ij} = \mathcal{S}_{i(i+1)} \cup \mathcal{S}_{(i+1)j} \setminus \{\sigma_i\}$

DÉMONSTRATION. Par la définition même du casseur, il est clair que $\mathcal{S}_{i(i+1)} \cup \mathcal{S}_{(i+1)j} \setminus \{\sigma_i\} \subseteq \mathcal{S}_{ij}$

Réciproquement, soit $x \notin I_{ij}$ tel que $x \notin \mathcal{S}_{i(i+1)} \cup \mathcal{S}_{(i+1)j}$. Par la propriété de transitivité de H_x, nous obtenons $H(x|yz)$ pour tout y et z de I_{ij}, ou en d'autres termes $x \notin \mathcal{S}_{ij}$. Par conséquent $\mathcal{S}_{ij} \subseteq \mathcal{S}_{i(i+1)} \cup \mathcal{S}_{(i+1)j}$. Finalement nous utilisons le fait que $\sigma_i \notin \mathcal{S}_{ij}$ pour conclure. □

Cela conduit à un algorithme naïf en temps $O(n^3)$ qui répond à la question considérée dans cette section. L'algorithme est le suivant, pour tous les intervalles I_{ij} nous calculons les ensembles $\mathcal{S}_{i(i+1)}$, ensuite nous calculons $\mathcal{S}_{ij}$ en s'appuyant des $\mathcal{S}_{(i+1)j}$ précédemment calculé et en utilisant la proposition 2.50. Il reste ensuite à tester si $\mathcal{S}_{ij}$ est vide. Si oui, I_{ij} est un module fort.

Nous améliorons maintenant cette idée de la manière suivante. Un intervalle I_{ij} est dit relaché à droite si il ne comporte pas de casseur à droite dans la permutation σ, plus formellement pour tout $k > j$, σ_k n'appartient pas à $\mathfrak{S}_{ij}$. Évidemment, si I_{ij} est un module fort de H, il est trivialement relaché à droite. Cependant un point de vue plus intéressant : si I_{ij} n'est pas relaché à droite, alors il n'y aura aucun $i' \leqslant i$ tel que $I_{i'j}$ soit une module fort. De plus :

PROPOSITION 2.51. *Si $j_1 < \ldots < j_k$ sont tels que pour tout $I_{ij_q}(1 \leqslant q \leqslant k)$ est relaché à droite alors nous avons $\mathfrak{S}_{ij_1} \subseteq \ldots \subseteq \mathfrak{S}_{ij_k}$*

DÉMONSTRATION. Tous les casseurs de ces intervalles sont sur la gauche de σ_i selon l'ordre de σ. Par conséquent un casseur s de I_{ij_q} ne peut pas appartenir à $I_{ij_{q+1}}$ mais appartiendra à $\mathfrak{S}_{ij_{q+1}}$. □

En quelques mots le principe de l'algorithme 3 est le suivant, si à une itération i $(1 \leqslant i \leqslant n)$, nous stockons seulement quelques intervalles relachés à droite dans une liste $RF = (I_{ij_1}, \ldots, I_{ij_k})$, et donc tous les casseurs correspondants peuvent être stockés par différence dans une liste $\Delta S = (\Delta_{j_1}, \ldots, \Delta_{j_k})$, où $\Delta_{j_1} = \mathfrak{S}_{ij_1}$ et $\Delta_{j_q} = \mathfrak{S}_{ij_q} \setminus \mathfrak{S}_{ij_{q-1}} (q \geqslant 2)$. Sous ce formalisme, un intervalle I_{ij_q} de la collection est un module si et seulement si les q premiers éléments de ΔS sont vides *i.e.* $\Delta_{j_1} = \cdots = \Delta_{j_q} = \varnothing$.

De l'itération i à $i-1$, la collection des intervalles sera développée de $RF = (I_{ij_1}, \ldots, I_{ij_k})$ à $RF = (I_{(i-1)(i-1)}, I_{(i-1)j_1}, \ldots, I_{(i-1)j_k})$ et la liste ΔS sera mise à jour selon les dispositions de la proposition 2.50.

De même si pour quelques j_q l'extension de I_{ij_q} à $I_{(i-1)j_q}$ introduit un casseur σ_k tel que $k > j_q$, alors nous supprimons cet intervalle de RF car il n'est plus relaché à droite et j_q n'a plus la moindre chance d'être la borne droite d'un module fort de H.

Nous en venons donc à l'algorithme 3. Afin d'alléger un tant soit peu les notations, nous désignerons les intervalles I_{ij_q} à l'itération i par leurs bornes à droite, ce qui donnera $RF = (j_1, \ldots, j_k)$.

Remarquons qu'à la première itération, à savoir $i = n$, il s'agit essentiellement d'une étape d'initialisation. En effet à la fin de la boucle, nous aurons toujours $RF = (n)$, $\Delta S = (\varnothing)$ et enfin $M = \{n\}$. Le "vrai" calcul commencera à l'itération suivante : quand $i = n - 1$.

INVARIANT 2.52. Pour tout $1 \leqslant i \leqslant n$ soit $RF_i = (j_1, \ldots, j_k)$ et $\Delta S_i = (\Delta_1, \ldots, \Delta_k)$ les valeurs de RF et ΔS à la fin de la première boucle **pour** (ligne 6) dans l'algorithme 3.

– Pour tout membre j de RF_i, l'intervalle I_{ij} est relaché à droite.
– Pour tout $1 \leqslant q \leqslant k$, $\mathfrak{S}_{ij_q} = \Delta_1 \cup \cdots \cup \Delta_k$.

La correction de l'algorithme 3 est une conséquence directe de l'invariant 2.52. Concernant la complexité, il est assez aisé de réaliser que le temps global passé dans toutes les

Entrées : Une relation homogène H sur l'ensemble X, et un permutation factorisante σ de H.
Sorties : L'arbre généralisé de décomposition modulaire $\mathcal{T}$ de H.

début
  $RF \leftarrow ()$
  $\Delta S \leftarrow ()$
  $M \leftarrow \varnothing$
  Créer un auxiliaire $y = \sigma(n+1)$ tel que $H(s|xy)$ pour tout $s \in X$ et $x = \sigma(n)$
  pour $i = n$ **à** 1 **faire**
    $x \leftarrow \sigma(i)$ et $y \leftarrow \sigma(i+1)$
    si x *appartient à un membre de* ΔS **alors**
      supprimer x de ce membre
    pour chaque $s = \sigma(l)$ *avec* $l < i$ *et* $\neg H(s|xy)$ **faire**
      Ajouter s à la liste des membres de ΔS
    Trouver $s = \sigma(r)$ tel que $\neg H(s|xy)$ et r maximum, sinon $r \leftarrow 0$
    tant que *le premier membre* j *de* RF *satisfait* $j < r$ **faire**
      Supprimer j de RF
      Fst et Snd sont le premier et le second membre de ΔS
      $Snd \leftarrow Snd \cup Fst$ et supprimer Fst de ΔS
    $RF \leftarrow (i, RF)$ de $\Delta S \leftarrow (\varnothing, \Delta S)$
    Soit S, *resp.* j, le premier membre de ΔS, *resp.* RF
    tant que $S = \varnothing$ **faire**
      $M \leftarrow \{I_{ij}\} \cup M$
      Soit S, *resp.* j, le membre suivant dans ΔS, *resp.* RF
  Supprimer les membres faibles de M
  Construire $\mathcal{T}$, l'arbre d'inclusion des membres de M
  retourner $\mathcal{T}$
fin

Algorithme 3 : Calcul de l'arbre de décomposition modulaire généralisé à partir d'une permutation factorisante.

boucles ne dépasse pas $O(n^2)$. Après ces boucles, supprimer les membres faibles de la liste M peut être fait en temps linéaire sur M en utilisant l'ordre lexical sur les membres de M : I_{ij} est avant $I_{i'j'}$ si et seulement si $i < i'$ ou $i = i'$ et $j \leqslant j'$. Notons que M est plus petit que le nombre d'intervalles de σ qui, on le rappelle, atteint $O(n^2)$. De la même manière ordonner les membres restants de M par ordre d'inclusion se fait en temps linéaire en utilisant l'ordre lexical du dessus. En conséquence de quoi, la complexité globale de l'algorithme 3 est en temps $O(n^2)$.

THÉORÈME 2.53. *Soit σ une permutation factorisante de H définie sur un ensemble X, nous pouvons calculer l'arbre de décomposition généralisé en temps $O(n^2)$.*

Une permutation factorisante peut être trouvée en $O(n^2)$ dans plusieurs cas, en particulier avec les relations homogènes standard de :

- Graphe d'héritage, une extension linéaire donne un telle permutation Ducournau et Habib (1989)
- Dans le cas des graphes chordaux (ou triangulés) renvoie cette permutation Hsu et Ma (1991).
- Dans les tournois, un algorithme simple d'affinage de partitions, (prendre un sommet x et couper en $N^-(x), \{x\}, N^+(x)...$) donne une permutation factorisante McConnell et Montgolfier (2005a,b) ;
- Dans le cas des graphes non-orientés, des algorithmes plus compliqués donne cette permutation en temps $O(m\ log\ n)$ Habib et al. (1999) ou en $O(n+m)$ Habib et al. (2004)

5. Bonnes relations homogènes

Par la suite nous appellerons *bonne* relation homogène une relation qui respecte la propriété du quotient modulaire (*cf.* section 2.3, définition 2.22). Afin de faciliter la lecture, nous rappelons ici la propriété du quotient modulaire : H est quotient modulaire si : $H(x|st) \Leftrightarrow H(y|st)$ pour tout module M de H, et pour tout $x, y \in M$ et $s, t \notin M$.

Par exemple, les relations homogènes standard sont de bonnes relations homogènes (*cf.* proposition 2.26). L'étude de ces bonnes relations homogènes est en particulier motivé par la proposition suivante :

PROPOSITION 2.54. *Les modules d'une bonne relation homogène forment une famille faiblement partitive.*

DÉMONSTRATION. Le lemme 2.12 fournit la clôture par union et par intersection de modules qui se chevauchent. Il reste à démontrer que nous obtenons également la différence. Soient A et B deux éléments de $\mathscr{M}_H$ tels que $A \oo B$ alors $A \setminus B$ est un module.

Considérons que $A \setminus B$ a un casseur s. Comme A est un module s appartient nécessairement à $A \cup B$. Considérons maintenant x et y deux éléments de $A \setminus B$ tel que $\overline{H(s|xy)}$. Comme $A \oo B$ il existe un élément t appartenant à $B \setminus A$ et comme B est un module, la propriété du quotient modulaire donne $\overline{H(t|xy)}$ et donc par conséquent A n'est pas un module, contradiction. □

Par la suite, et étant donné une bonne relation homogène H définie sur l'ensemble X, nous étudions le problème de trouver l'arbre de décomposition de H, à savoir l'arbre d'inclusion de modules forts de H. Une fois encore, l'algorithme de la section 4.4 fournit l'arbre

recherché en temps $O(n^3)$. Cependant, dans cette section, en s'appuyant sur les propriétés structurelles fortes des bonnes relations homogènes, nous présentons un algorithme plus efficace. Son temps d'exécution est $O(n^2)$ autant dire en la taille de la donnée quand la relation homogène est codée sous forme de matrice. La solution proposée ici s'inspire des travaux de Ehrenfeucht et al. (1994) sur les 2-structures.

Tout d'abord il est nécessaire d'introduire quelques notions supplémentaires.

Définition 2.55. Un super arbre de décomposition modulaire (SMDT) d'un bonne relation homogène H définie sur X est un arbre $\mathcal{T}$ qui satisfait les propriétés suivantes :

- Les feuilles de $\mathcal{T}$ sont étiquetées par X.
- Chaque noeud de l'arbre représente un module de H.
- Chaque module fort de H est un noeud de $\mathcal{T}$.

L'idée de l'algorithme est la suivante, cela consiste à calculer une branche gauche (une chenille plus connue sous le nom de *"caterpillar"*) du super arbre de décomposition modulaire de H, partant de la racine X jusqu'à un élément arbitraire x (*cf.* Fig. 2.8). Par la suite, l'algorithme calcule récursivement les jambes ("legs") de la chenille, et les rattache à la chenille.

L'algorithme 4 retranscrit cette idée. Par la suite, le super arbre de décomposition modulaire est typé en arbre de décomposition modulaire.

Entrées : Une bonne relation homogène H définie sur l'ensemble fini X
Sorties : Un super arbre de décomposition modulaire $\mathcal{T}$ de H

début
  x un élément de X
  Calculer $MaxM(x)$, les modules maximaux ne contenant pas x
  Ordonner $MaxM(x) = M_1, \ldots, M_k$ tel que pour chaque $B_j \in \mathcal{B}$, $1 \leq j \leq l$, il existe $f(j)$ tel que $B_j = \{x\} \uplus M_1 \uplus M_2 \uplus \cdots \uplus M_{f(j)}$
  Initialiser $\mathcal{T}$ comme $x-$branche
  pour chaque M_i $(1 \leq i < k)$ **faire**
    Calculer récursivement le SMDT $\mathcal{T}_i$ de $H[M_i]$
    Ajouter $\mathcal{T}_i$ au noeud B_j de $\mathcal{T}$ tel que $j \leq i < f(j)$
  retourner $\mathcal{T}$
fin

Algorithme 4 : Super Arbre de Décomposition Modulaire d'une Bonne Relation Homogène

Proposition 2.56. *L'algorithme 4 calcule un super arbre de décomposition modulaire.*

Démonstration. De manière évidente tous les noeud renvoyés sont des modules. Nous devons seulement vérifier que l'arbre contient tous les modules forts de H. Ceci est vrai bien

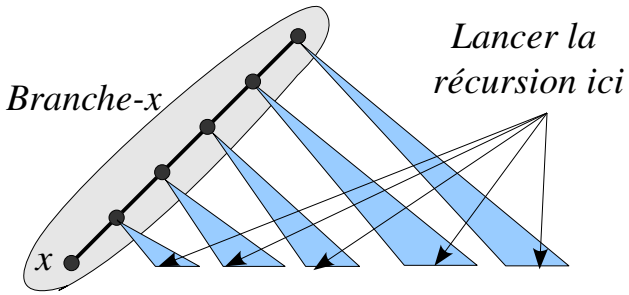


Figure 2.8. Une approche récursive pour calculer le super arbre de décomposition modulaire.

sûr, car pour un module fort M de H, le premier élément $x \in M$ pris pour la branche$-x$ (*cf.* définition 2.57) à quelque étape récursive que ce soit, renvoie $M \in \mathcal{B}(x)$. Le fait que la relation H soit une bonne relation vérifie la propriété suivante : quand l'algorithme est appelé récursivement sur $H[N]$ et quand N est un module, le module M de $H[N]$ renvoyé par l'algorithme est exactement le module M de H. □

5.1. Modules forts contenant x.

Définition 2.57 (Branche$-x$). La branche$-x$ d'une bonne relation homogène H définie sur l'ensemble X est l'ensemble $\mathcal{B}(x)$ de tous les modules forts qui contiennent l'élément $x \in X$, ordonnés par inclusion. En d'autres termes, il s'agit d'un chemin de la racine (X) à la feuille x de l'arbre de décomposition modulaire de H.

La technique utilisée pour construire les modules forts contenant x utilise la technique présentée à la section 4.2 avec l'algorithme 2 $MaxM$ qui calcule tous les modules forts maximaux ne contenant pas x. À la section 4.2 nous avons défini $MaxM(x)$ comme étant la partition de $X \setminus \{x\}$ en $\{M_1, \ldots, M_k\}$ modules maximaux ne contenant pas x. Regardons maintenant quelles sont les relations qui existent entre $MaxM(x)$ et $\mathcal{B}(x)$.

Proposition 2.58. *Les modules de* $MaxM(x) = \{M_1, \ldots, M_k\}$ *peuvent être ordonnés de* 1 *à* k *de la manière suivante :*
Pour chaque $B \in \mathcal{B}(x)$ *il existe* f *tel que* $B = \{x\} \uplus M_1 \uplus M_2 \uplus \cdots \uplus M_f$.

Démonstration. Pour un module $B \in \mathcal{B}(x)$, les modules maximaux ne contenant pas B forment une partition de X. Bien sûr chaque module maximal de $MaxM(x)$ est inclus (ou égal) dans un module de cette partition. Par conséquent un module de $MaxM(x)$ ne peut pas chevaucher $B \in \mathcal{B}(x)$. Ensuite, concernant l'ordre des modules, il suffit de numéroter les modules de $\mathcal{B}(x)$ en partant de $B_0 = \{x\}$ jusqu'à $B_l = X$ en utilisant l'ordre

d'inclusion. Ensuite il faut numéroter les modules de $MaxM(x)$ inclus dans B_1 de 1 à $f(1)$, les modules de $MaxM(x)$ inclus dans B_2 mais pas dans B_1 de $f(1)+1$ à $f(2)$ et de manière plus générale les modules de B_i qui ne sont pas contenus dans B_{i-1} de $f(i-1)+1$ à $f(i)$. □

Une conséquence de cela, est que si nous ordonnons les éléments de la branche$-x$ de $B_0 = \{x\}$ à $B_l = X$ par ordre croissant d'inclusion, alors pour tout $1 \leqslant i \leqslant l$, $B_{i+1} \setminus B_i$ est égal aux éléments de $MaxM(x)$ qui se suivent selon l'ordre défini ci-dessus.

PROPOSITION 2.59. *Soit $B_i \in \mathcal{B}(x)$ un module fort contenant x qui ne soit pas une feuille, et soient $C_i^1, \ldots, C_i^{g(i)}$ ses enfants dans l'arbre de décomposition modulaire. Soit j tel que $C_i^j = B_{i+1}$ est l'enfant contenant x. Si B_i est linéaire nous supposons que ses enfants sont ordonnés selon l'ordre linéaire.*

- *Si B_i est premier alors pour tout $k \neq j$ $C_i^k \in MaxM(x)$.*
- *Si B_i est linéaire alors $\cup_{k=1}^{j-1} C_i^k \in MaxM(x)$ et $\cup_{k=j+1}^{g(i)} C_i^k \in MaxM(x)$.*
- *Si B_i est complet alors $\cup_{k \neq j} C_i^k \in MaxM(x)$.*

Il n'y a pas d'autre type d'élément dans $MaxM(x)$ que ceux décrits au dessus.

5.2. Relation quotient.

Construisons maintenant la relation quotient. Pour chaque M_i de $MaxM(x)$, soit e_i, élément de M_i, un élément représentatif de M_i (n'importe quel élément de M_i). La relation quotient de H par $MaxM(x)$, notée par la suite $H(x)$, est la relation :

$$H(x) = H[\{x, e_1, \ldots, e_k\}]$$

PROPOSITION 2.60. *La relation quotient de H par $MaxM(x)$ ne dépend pas du choix des représentants pour chaque M_i.*

DÉMONSTRATION. Cela est dû au fait que la relation homogène est une bonne relation homogène. □

PROPOSITION 2.61. *Chaque module non trivial de $H(x)$ contient x.*

DÉMONSTRATION. Procédons par l'absurde et considérons qu'il existe un module $\cup_{i \in I}\{e_i\}$ de $H(x)$ qui ne contienne pas x. Par conséquent, nous avons $|I| \geqslant 2$, et $\cup_{i \in I} M_i$ est un module de H qui ne contient pas x, et est plus grand qu'un élément de $MaxM(x)$, d'où une contradiction. □

Pour $M_i \in MaxM(x)$, définissons $S(M_i) \in \mathcal{B}(x)$ comme étant le plus petit module de $\mathcal{B}(x)$ qui contient M_i. En utilisant les notations de la proposition 2.58 si $S(M_i) = B_j$ alors $i \leqslant j \leqslant j(i)$. La proposition 2.60 fournit la relation qu'il y a entre M_i et $S(M_i)$ en fonction du type de $S(M_i)$ (à savoir complet, linéaire ou premier). Nous dirons que $e_i \in M_i$ est un élément$-P$ (*resp.* élément$-L$, élément$-C$) si $S(M_i)$ est premier (*resp.* linéaire ou

complet). Soient e_i et e_j des éléments représentatifs respectivement de M_i et de M_j, ces deux éléments sont dits *compagnons* l'un de l'autre si $S(M_i) = S(M_j)$.

La proposition 2.60 nous dit que e_i n'a pas de compagnon si $S(M_i)$ est complet. Il a au plus un compagnon si $S(M_i)$ est linéaire et au moins un si $S(M_i)$ est premier.

5.3. Graphe de forçage.

DÉFINITION 2.62 (Graphe de Forçage). En conservant les notations introduites ci-dessus, nous définissons le graphe de forçage associé à x comme :

$G(x) = (V, A)$ est défini comme $V = \{e_1, \ldots, e_k\}$ et un arc (e_i, e_j) appartient à A si et seulement si $\overline{H(e_j|x, e_i)}$.

PROPOSITION 2.63. *Soit y un sommet de $G(x)$ et soient $N^*(y)$ les descendants de y dans $G(x)$ (y inclus). Nous obtenons que $N^*(y) \cup \{x\}$ est le plus petit module de $H(x)$ contenant y.*

DÉMONSTRATION. Remarquons tout d'abord que tous les modules non triviaux de $H(x)$ contiennent x (*cf.* 2.61). Maintenant, si le graphe de forçage admet une arête (e_i, e_j) alors tout module non trivial de $H(x)$ contenant e_i doit également contenir e_j. En conséquence de quoi tous les descendants de y dans $G(x)$ sont dans tous les modules contenant y et x.

Nous devons maintenant montrer que pour tout sous-ensemble de sommets A de $G(x)$ n'ayant aucun arc sortant de A, $A \cup \{x\}$ est un module de $H(x)$.

Par définition de $G(x)$, pour tout $u \in A$ et pour tout $v \notin A$ nous avons $H(v|x, u)$. De plus comme la relation H est transitive, pour tout u, u' de A nous avons $H(v|uu')$ et par conséquent $A \cup \{x\}$ est un module de $H(x)$. Donc finalement $N^*(y) \cup \{x\}$ est un module de $H(x)$ et c'est le plus petit à contenir à la fois x et y. □

Soit C une composante fortement connexe (CFC pour faire court) de $G(x)$. La proposition précédente nous dit que tous les sommets de C sont compagnons. De plus nous avons :

PROPOSITION 2.64. *Une composante fortement connexe non triviale de $G(x)$ est formée par des compagnions qui sont éléments$-P$.*

Réciproquement un ensemble maximal de compagnons éléments$-P$ est fortement connexe.

DÉMONSTRATION. Selon la proposition 2.59 il n'y a pas de compagnons qui soient des éléments$-C$, et il y a au plus deux compagnons qui soient des éléments$-L$. Mais clairement il n'y a pas d'arc entre eux. Par conséquent une CFC avec au moins deux sommets contient des compagnons qui sont éléments$-P$. Selon la proposition 2.63 si des éléments$-P$ compagnons avaient été coupé en deux (ou plus) CFCs C et D, alors il y aurait

soit un module de $H(x)$ qui contiendrait C mais pas D, soit un module qui contiendrait D mais pas C. Dans les deux cas, le plus petit module de $H(x)$ qui contient $C \cup D$ ne peut pas être premier. □

PROPOSITION 2.65. *Deux compagnons éléments−L sont des faux jumeaux (ils ont le même voisinage et il n'y a pas d'arc entre eux).*

Réciproquement les paires de faux jumeaux sont exactement des compagnons éléments−L.

DÉMONSTRATION. Soient e et e' deux compagnons éléments−L. Le plus petit module M de $H(x)$ contenant à la fois e et e' est par conséquent un module linéaire $\{e\} \cup M' \cup \{e'\}$ où M' est un module fort enfant de M dans l'arbre de décomposition modulaire de $H(x)$. Bien sûr $x \in M'$. À la fois $\{e\} \cup M'$ et $M' \cup \{e'\}$ sont des modules, et les descendants de e sont exactement ceux de e' et ils sont les éléments de M', en accord avec la proposition 2.63. De plus comme H est une bonne relation homogène, e et e' sont jumeaux. □

En conséquences des propositions 2.63, 2.64 et 2.65 nous obtenons :

PROPOSITION 2.66. *Toute extension linéaire (tri topologique) de $G(x)$ ordonne $MaxM(x)$ selon l'ordre défini dans la proposition 2.58.*

PROPOSITION 2.67. *La branche−x de H peut être calculée en temps $O(Q(MaxM(x)))$.*

DÉMONSTRATION. Nous rappelons que $Q(\mathcal{P})$ est le nombre de paires $\{x, y\}$ d'éléments qui ne sont pas dans une même partie de la partition $\mathcal{P}$ (*cf.* définition 2.44). Soit k le nombre de parties de la partition $MaxM(x)$. Trivialement $k^2 = O(Q(MaxM(x)))$ et $Q(MaxM(x)) = O(n^2)$.

L'algorithme est le suivant :

- Les modules maximaux ne contenant pas x ($MaxM(x)$) peuvent être calculés en temps $O(Q(MaxM(x)))$, selon le théorème 2.45 en utilisant l'algorithme 2.
- Par la suite les sommets du graphe de forçage sont déterminés de manière arbitraire : pour tout $1 \leqslant i \leqslant k$, $e_i \in M_i$.
- Ensuite, la construction du graphe de forçage $G(x)$ en temps $O(k^2)$ est facile.
- Le calcul d'une extension linéaire de $G(x)$ se fait en temps $O(k^2)$ sans difficulté.
- Au final la proposition 2.58 nous dit comment construire $\mathcal{B}(x)$ à partir de l'ordre de $MaxM(x)$. Notons que tous les sommets *compagnons* apparaissent consécutivement dans l'extension linéaire et sont tous regroupés pour former $B_{i+1} \setminus B_i$.

□

Nous obtenons donc :

THÉORÈME 2.68. *L'algorithme 4 calcule un super arbre de décomposition en temps $O(n^2)$.*

DÉMONSTRATION. C'est une application directe des propositions 2.56 et 2.67. Nous devons tout de même montrer que la somme de tous les $O(Q(MaxM(x)))$ est en $O(n^2)$.

Cela est vrai car $Q(MaxM(x))$ est le nombre de paires $\{x, y\}$ qui appartiennent à des parties différentes de $MaxM(x)$. Comme l'algorithme est lancé récursivement sur les module de $MaxM(x)$, chaque paire $\{x, y\}$ n'est comptée qu'une seule fois. □

5.4. Suppression des modules faibles et typages des noeuds.

Une fois que nous avons obtenu le super arbre de décomposition $\mathcal{T}$, toutefois cet arbre n'est pas complètement un arbre de décomposition modulaire, en effet il peut subsister des modules faibles et les noeuds internes ne sont pas typés (*i.e.* premiers, complets ou enfin linéaires).

DÉFINITION 2.69 (Quotient d'un Noeud). Soit N un noeud du super arbre de décomposition $\mathcal{T}$ de H, avec pour enfants $S_1, \ldots, S_k$ et e_i un élément arbitraire. Le quotient de H par N est $H[\{e_1, \ldots, e_k\}]$.

PROPOSITION 2.70. *La relation quotient d'un noeud N du super arbre de décomposition est soit :*

- *De type P : avec aucun module non-trivial.*
- *De type L : les éléments peuvent être ordonnés de manière linéaire de telle manière que les modules de la relation quotient soient exactement les intervalles de la relation.*
- *De type C : tout sous-ensemble est un module.*

Si le noeud N a k enfants, un algorithme trivial (en temps $O(k^2)$) peut tester le type et ordonner les éléments si nécessaire.

Un résultat classique et pas trop difficile à démontrer est le suivant :

PROPOSITION 2.71. *Soit T un arbre avec n feuilles et aucun noeud (interne) avec un seul enfant, alors :*

$$\sum_{N \text{ noeud de } T} d(N)^2 = O(n^2)$$

Nous pouvons donc effectuer un calcul en temps quadratique sur chaque noeud de $\mathcal{T}$. Une application de la proposition 2.71 est la

PROPOSITION 2.72. *Soit H une bonne relation homogène définie sur X. Calculer la relation quotient pour tous les noeuds du super arbre de décomposition $\mathcal{T}$ de H se fait en temps $O(n^2)$.*

DÉMONSTRATION. Il suffit d'utiliser une approche "bottom-up" sur le super arbre, et ne conserver qu'un seul représentant par enfant. Chaque relation quotient peut donc être calculée en temps linéaire en sa taille *i.e.* en temps $O(k^2)$. □

Une autre application de la proposition 2.71 combinée à la proposition 2.70 nous donne le résultat suivant : le typage des noeuds (internes) du super arbre de décomposition peut se faire en temps $O(n^2)$.

Remarquons que nous nous autorisons à typer des modules faibles. Par la suite, nous recherchons les modules faibles et nous transformons le super arbre de décomposition en un arbre de décomposition modulaire, et ce en utilisant la proposition suivante.

PROPOSITION 2.73. *Soit H une bonne relation homogène définie sur X, et N un noeud du super arbre de décomposition $\mathcal{T}$ et soit F son parent dans $\mathcal{T}$. Par ailleurs, comme $\mathcal{T}$ est super arbre de décomposition, F a un autre enfant que nous appellerons A. Si F est de type linéaire alors, nous prenons A qui précède immédiatement (ou suit) N dans l'ordre linéaire. Nous prenons un élément a de A. Si N est non-trivial il a moins deux enfants B et C. Si N est linéaire alors nous prenons pour B son premier enfant et pour C son dernier. Finalement prenons b (resp. c) un élément de B (resp. C). Alors :*

N est un module faible si et seulement si

$$\{a, b\} \text{ ou } \{a, c\} \text{ est un module de } H[\{a, b, c\}].$$

DÉMONSTRATION. Si $\{a, b\}$ ou $\{a, c\}$ est un module alors N est trivialement un module faible.

Réciproquement si N est faible cela signifie qu'il est chevauché par un autre module N'. Par conséquent N et N' ont le même parent F dans l'arbre de décomposition modulaire. Si F est complet toute union d'enfants de F est incluse dans N et non incluse dans N' qui chevauche N. Comme $b \in N$ et $a \in F \setminus N$ nous obtenons le résultat désiré.

Si maintenant F est de type linéaire (tout autre arc est exclus), alors soit le premier enfant de F inclus dans N plus le précédent selon l'ordre linéaire, chevauche N, et $\{a, b\}$ est un module, ou le dernier enfant de F est inclus dans N plus le suivant selon l'ordre linéaire, chevauche N, et $\{a, c\}$ est un module. □

Cette proposition combinée avec la proposition 2.71, nous dit que les modules faibles peuvent être supprimés du super arbre de décomposition en temps $O(n^2)$. Nous obtenons finalement

PROPOSITION 2.74. *Un super arbre de décomposition modulaire de H peut être typé (transformé) en un arbre de décomposition modulaire en un temps $O(n^2)$.*

Avec la proposition précédente associée au théorème 2.68 nous obtenons :

THÉORÈME 2.75. *L'arbre de décomposition modulaire d'une bonne relation homogène H définie peut être trouvé en temps $O(n^2)$.*

Conjecture : *Lorsque la relation homogène H est donnée sous forme de listes d'"adjacence", l'arbre de décomposition peut être construit en temps $O(n+m)$, où $n = |X|$ et m la somme des tailles des listes.*

6. Bilan et perspectives

Un petit bilan de ce qui a été présenté sur les relations homogènes s'impose. Regardons quelles sont les conséquences de cette théorie sur la décomposition des graphes, des 2-structures et autres relations.

De la proposition 2.27 de la section 2.4 nous apprenons que la famille des modules des graphes non-orientés et des 2-structures symétriques forme une famille partitive, alors que la famille des modules d'un graphe orienté forme une famille faiblement partitive.

Dans le cas des graphes nous pouvons définir d'autres relations homogènes que la relation standard, en conservant le paradigme fort de la distinction et s'abstraire complètement de la notion de voisinage.

Nous présentons ici quelques unes de ces relations. Par exemple la relation définie comme suit est homogène *«Il existe un chemin de x à y évitant s»* ou de manière plus générale *«Il existe un chemin de x à y évitant le voisinage de s»*. Il est relativement aisé de voir que ces deux relations satisfont les conditions de base (*cf.* définition 2.1 *p.* 23) d'une relation homogène, à savoir symétrie, réflexivité et enfin transitivité. En ce qui concerne la première relation, les modules forment une partition. Il s'avère que la partition ainsi obtenue correspond au composantes 2-connexes du graphe, toutefois les points d'articulations n'apparaissent pas. Pour la seconde relation, les modules obtenus sont assez proche de la décomposition en star cutset introduite par Chvátal (1985).

Une autre relation intéressante est la relation homogène de distance noté $D_k(s|xy)$ si $d(s,x) \leqslant k$ et $d(s,y) \leqslant k$, où $d(x,y)$ représente la distance (longueur du plus court chemin) de x à y. Dans le cas où $k = 1$, la relation est exactement la décomposition modulaire (pour cause d'adjacence). Cela pourrait être intéressant de considérer le cas général.

Remarquons finalement que malgré une définition abstraite et assez simple la plupart des algorithmes présentés admettent une complexité linéaire ou quasi linéaire et même si la définition ne nous le laissait pas présager.

CHAPITRE 3

Umodules

Le but de ce chapitre est d'introduire un nouveau cadre de décomposition sur les relations homogènes. Cette nouvelle décomposition se nomme umodule, où *"u"* signifie *"unordered"* ou *"unsigned"* module.

Afin de relaxer la notion de modules sur les relations homogènes, il nous est apparu naturel de considérer le point de vue dual, à savoir, se placer à l'intérieur et non plus à l'extérieur de l'ensemble considéré. En effet dans le cas des modules, ce sont les éléments extérieurs qui agissent de manière similaire vis à vis du module. Ce faisant il est aisé de quotienter le module par un sommet représentant. Dans ce chapitre nous envisageons le cas antagoniste, à savoir, un umodule, est un ensemble qui distingue l'extérieur de la même façon. Nous présenterons une définition formelle dans la suite du manuscrit. Une des principales conséquences de ce point de vue est que cela permet sur certaines structures, en particulier les graphes, de généraliser la décomposition modulaire. Cependant nous montrerons que ce n'est pas le cas pour une relation homogène quelconque.

Dans un premier temps nous présenterons les définitions des umodules, et nous étudierons quels sont les graphes où les modules et les umodules coïncident (Section. 1), nous considérerons ensuite les aspects algorithmiques :

comment trouver des umodules forts à l'aide des techniques d'affinage de partitions (Section. 2). Nous présenterons ensuite un algorithme en $O(n^5)$ pour calculer l'arbre de décomposition umodulaire d'une relation homogène quelconque.

Nous nous intéressons ensuite (Section. 3) aux relations homogènes particulières qu'il est possible d'obtenir plus d'informations, notamment sur la structure de la famille des umodules, et ainsi obtenir des algorithmes plus performant. En effet nous obtiendrons un algorithme en $O(n^3)$ pour les relations homogènes de congruence locale égale à 2, et un algorithme en $O(n^2)$ pour les relations homogènes dites auto-complémentées.

Nous présenterons (Section. 4) une opération qui modifie localement la relation homogène, et ce faisant permet de grandement simplifier le problème en se ramenant à un

problème de décomposition modulaire. Cette transformation se nomme le *Seidel switch*, nous présenterons quelques une de ses propriétés.

Par la suite nous montrerons (Section. 5) à quoi correspondent les umodules sur les graphes non-orientés, quelle est la famille des graphes totalement décomposables. Nous verrons (Section. 6) également comment cette décomposition fournit une nouvelle manière de décomposer les tournois. Nous montrerons que la classe des tournois localement transitifs est également la classe des tournois qui sont totalement décomposables par la décomposition umodulaire. Par la suite nous évoquerons des problèmes algorithmiques reliés aux tournois, et nous fournirons un algorithme linéaire pour tester l'isomorphisme de deux tournois localement transitifs ainsi qu'un algorithme linéaire pour résoudre le problème du *Feedback Vertex Set* sur cette classe.

Les résultats obtenus dans ce chapitre ont donné lieu aux plublications suivantes : Bui-Xuan et al. (2007) et Bui-Xuan et al. (2008b).

Plan

1. Umodules : définitions

DÉFINITION 3.1 (Umodule). Soit X un ensemble fini et H une relation homogène définie sur X. Un umodule de H est un sous ensemble U de X tel que :

$$\forall u, u' \in U \text{ et } \forall x, x' \in X \setminus U : H(u|xx') \Longleftrightarrow H(u'|xx')$$

En d'autres termes, les éléments de l'umodule voit l'extérieur de la même façon à condition que l'on oublie la *"signature"* de la partition. Par exemple dans le cas des graphes non orientés, pour chaque sommet x de X la partition associée comporte au plus deux classes. Dans un graphe un umodule peut soit être constitué de jumeaux ou d'anti-jumeaux...

De la même manière que pour les modules, on peut considérer des umodules comme triviaux, à partir du moment où l'ensemble ne comporte qu'un seul élément ou lorsqu'il en contient au moins $n-1$ (où n est la taille de l'ensemble support X).

.

PROPOSITION 3.2 (Clôture par Union). *Pour tout umodules U, U' d'une relation homogène H, si $U \cap U' \neq \emptyset$. alors $U \cup U'$ est aussi un umodule de H.*

DÉMONSTRATION. Soit v un élément de $U \cap U'$. Pour tout élément u de U et pour tout élément u' de U' et pour tout $x, x' \notin U \cup U'$ nous avons :

$$H(u|xx') \Longleftrightarrow H(v|xx') \Longleftrightarrow H(u'|xx')$$

Pour ce faire, nous nous appuyons sur la transitivité de la relation H. □

PROPOSITION 3.3. *Soit H une relation homogène sur l'ensemble X. Si H est une relation homogène graphique (i.e.. celle d'un graphe) alors les modules de H sont aussi des umodules.*

DÉMONSTRATION. Direct par la définition 3.1 et 2.6 □

PROPOSITION 3.4. *Soit H une relation homogène sur l'ensemble X. Si H est une relation homogène quelconque alors pour tout module M de H, $X \setminus M$ est un umodule de H.*

DÉMONSTRATION. Remarquons que si $X \setminus M$ n'est pas un umodule, il existe alors deux éléments m et m' de M et deux éléments x et x' de $X \setminus M$. En appliquant la définition 3.1 cela se traduit par, sans perte de généralité,

$$H(x|mm') \text{ et } \overline{H(x'|mm)}$$

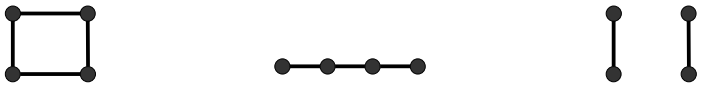
et donc M n'est pas un module. □

REMARQUE 3.5. Toutes les propriétés des modules ne sont pas conservées, en effet nous avons vu que si M est un module de H et $M' \subseteq M$ alors M' est un module de H si et seulement si M' est un module de $H[M]$. Cette propriété ne demeure plus vraie pour les umodules.

1.1. Les graphes de thresholds.

Quand les umodules rencontrent les modules...

DÉFINITION 3.6 (Threshold graph Brandstädt et al. (1999)). Les graphes de threshold sont obtenus à partir d'un sommet isolé, en ajoutant successivement des sommets dominants (*i.e.* relié à tous les sommets du graphes) et des sommets isolés. Une définition équivalente en terme de sous-graphes interdits est : un graphe est un graphe threshold si et seulement si il ne contient pas de C_4 le cycle à 4 sommets, le P_4 le chemin à quatre sommets, et $2K_2$ deux arêtes disjointes comme sous graphe induits. D'autres définition sont proposées dans Brandstädt et al. (1999)



(a) C_4 (b) P_4 (c) $2K_2$

Figure 3.1. Les sous graphes induits interdits des graphes Threshold.

PROPOSITION 3.7. *Un graphe G est un graphe de Threshold si et seulement si dans tous les sous graphes induits H de G, tout umodule de H est soit un module ou le complémentaire d'un module ou les deux.*

DÉMONSTRATION. ($\Leftarrow$) Si G n'est pas un graphe de threshold, cela signifie qu'il contient au moins un sous-graphe induit G' qui est isomorphe soit à P_4, C_4 ou co-C_4 (*cf.* figure 3.1). Dans tous les cas G' contient un umodule composé de deux sommets qui n'est pas un module ni le complémentaire d'un module (dans le cas du P_4 les deux sommets non adjacent, pour le C_4 toute paire de sommets reliés par une arête, enfin pour le co-C_4 toute paire de sommets non adjacent).

($\Rightarrow$) Réciproquement, soit $U \subseteq V'$ un umodule d'un sous-graphe induit $G' = (V', E')$ de G tel que U est ni un module, ni le complémentaire d'un module de G'. Soit $W = V' \setminus U$. Le fait que U ne soit pas le complémentaire d'un module implique l'existence d'un élément $a \in U$ et des éléments $b, c \in W$ tels que a est un casseur de $\{b, c\}$ (*i.e.* $\overline{H(a|bc)}$, où H est relation standard de G). Sans perte de généralité, nous suppons que $ab \in E$ et $ac \notin E$. Soit A l'ensemble qui contient tous les voisins de b qui appartiennent à U et $D = U \setminus A$. Soit B l'ensemble de tous les voisins de a qui appartiennent à W et $C = W \setminus B$ (*cf.* figure 3.2 pour la construction). En utilisant le fait que U soit un umodule, et que $a \in A$, $b \in B$ et $c \in C$, nous pouvons déduire que pour tout $x \in A$, $y \in B$, $z \in C$ et $t \in D$ tels que xy et zt soient des arêtes et que xz et yt n'en soient pas (ce qui correspond aux bi-joins qui seront traités en section 5 *p.* 80).

De plus comme U n'est pas un module (car $\overline{H(c|ta)}$), nous pouvons déduire qu'il existe au moins un sommet d dans D. Finalement il est facile de voir que $G[\{a, b, c, d\}]$ est soit un P_4, soit un C_4 ou enfin un co-C_4 (cela dépend seulement de la présence des arêtes entre A et B et entre C et D). □

REMARQUE 3.8. On remarquera que $P_4 = \overline{P_4}$ et que $C_4 = \overline{2K_2}$.

Les graphes de Threshold sont connus pour être une des plus petite classes de graphes (on pourra consulter les ouvrages de Brandstädt et al. (1999) et de Mahadev et Peled (1995)). Cependant pour la pluspart des graphes, les modules et les umodules diffèrent. La section 5 *p.* 80 sera dédiée à cette décomposition sur les graphes.

Figure 3.2. Construction de la preuve de la proposition 3.7

Toutefois avant d'explorer les formidables propriétés structurelles qu'offrent les umodules, intéressons nous d'abord à leur calcul.

2. Aspects algorithmiques

Lorsque nous avons commencé nos travaux sur les umodules, nous nous posions la question de savoir s'il existait une représentation compacte, *i.e.* polynomiale en la taille de l'ensemble support (ici X), de la famille des umodules dans le cas général. Rappelons que dans le cas général, quand la relation homogène H n'a pas de structure particulière, la famille des umodules $\mathscr{U}_H$ est fermée pour l'union d'éléments qui se chevauchent. Il est montré dans la thèse de Bui-Xuan (2008) qu'il n'y a pas d'espoir de coder cette famille en espace polynomial en n (*i.e.* $O(n^k)$, k une constante positive). Ce résultat consitue certes une mauvaise nouvelle, cependant il est tout de même possible de calculer certaines propriétés intéressantes sur les umodules.

Les premiers objets qui semblent intéressants à calculer sont les umodules maximaux vis à vis d'un coupe. En utilisant cela, nous fournissons également un algorithme polynomial pour calculer les umodules forts (*cf.* définition ci-après).

2.1. Umodules maximaux vis à vis d'une coupe.

Concernant les partitions, nous utilisons les même notations que celles utilisées aux Chapitre 2.

Soit S un sous-ensemble de X. Comme la famille des umodules est close par union d'éléments qui s'intersectent (*cf.* proposition 3.2) les umodule maximaux vis à vis de l'inclusion qui sont soit dans S soit dans $X \setminus S$ forment une partition de X, que nous noterons $MU(S) = MU(X \setminus S)$. En d'autres termes, il s'agit de la partition la plus épaisse des umodules de H qui est plus fine que la partition $\{S, X \setminus S\}$. De manière plus générale, cette partition donne une indication sur la disposition les umodules maximaux vis à vis de

S. Un umodule est soit inclus dans S,soit dans $X \setminus S$, soit il intersecte proprement S (*i.e.* chevauche) ou est trivial.

DÉFINITION 3.9. Soit H une relation homogène définie sur l'ensemble fini X. Soit C un sous-ensemble de X. La relation R_C sur C est définie comme :

$$\forall x, y \in C,\ R_C(x, y) \text{ si } \forall a, b \in (X \setminus C),\quad H(x|ab) \iff H(y|ab)$$

Il s'agit clairement d'une relation d'équivalence sur C. De plus, C est un umodule de H si et seulement si R_C ne comporte qu'une seule classe d'équivalence. Nous définissons maintenant une règle d'affinage, l'outil algorithmique principal pour calculer $MU(S)$.

DÉFINITION 3.10. Soit $\mathcal{P}$ une partition de X, et C une partie de $\mathcal{P}$. Soient $C_1, \ldots, C_k$ les classes d'équivalences de R_C. **RefineSelf**$(\mathcal{P}, C)$ est la partition obtenue à partir de $\mathcal{P}$ en remplaçant C par $C_1, \ldots, C_k$.

LEMME 3.11. *Soit H une relation homogène définie sur X, U est un umodule de H et soit $\mathcal{P}$ une partition de X.*

(1) *Si U est inclus dans une partie de $\mathcal{P}$, alors pour toute partie C de $\mathcal{P}$, U est inclus dans une partie de* **RefineSelf**$(\mathcal{P}, C)$.

(2) *Une partie C de $\mathcal{P}$ est un umodule si et seulement si $\mathcal{P} =$* **RefineSelf**$(\mathcal{P}, C)$.

DÉMONSTRATION. Soit U un umodule de H et soit P la partie qui contient U. En ce qui concerne le premier point, considérons $\mathcal{Q} =$ **RefineSelf**$(\mathcal{P}, C)$, où C est une partie de $\mathcal{P}$. Si $P \neq C$, alors P demeure une partie de $\mathcal{Q}$ et contient toujours U. Sinon si $P = C$, alors comme U est un umodule, les éléments de U ne peuvent être séparés en utilisant l'affinage.

La preuve du second point est immédiate car C est un umodule si et seulement si $\mathcal{P} =$ **RefineSelf**$(\mathcal{P}, C)$. □

Nous pouvons proposer maintenant un algorithme pour calculer $MU(S)$ en itérant la procédure **RefineSelf**.

Entrées : S un sous-ensemble de X, et H une relation homogène
Sorties : $MU(S)$

début
  $\mathcal{P} \leftarrow \{S, X \setminus S\}$
  tant que *Il existe une partie C non marquée dans $\mathcal{P}$* **faire**
    si $\mathcal{P} =$ **RefineSelf**$(\mathcal{P}, C)$ **alors**
      Marquer C
    sinon
      $\mathcal{P} \leftarrow$ **RefineSelf**$(\mathcal{P}, C)$
  $MU(S) \leftarrow \mathcal{P}$
  retourner $MU(S)$
fin

Algorithme 5 : Algorithme d'affinage qui calcule la partition $MU(S)$

THÉORÈME 3.12. *L'algorithme 5 calcule $MU(S)$ en temps $O(n^3)$.*

DÉMONSTRATION. Grâce au point 1 du lemme 3.11, à toute étape de l'algorithme l'invariant suivant reste vrai : $MU(S) \leqslant \mathcal{P} \leqslant \{S, X \setminus S\}$. Et grâce au point 2 du même lemme, le processus se termine lorsque $\mathcal{P} = MU(S)$. Les parties marquées sont les umodules.

Afin de respecter la complexité annoncée, chaque partie de la partition courante peut être marquée. Une partie devient marquée, si elle n'affine pas la partition. Lorsque $\mathcal{P}$ est remplacée par $RefineSelf(\mathcal{P}, C)$, les parties marquées de **RefineSelf**$(\mathcal{P}, C)$ sont exactement les parties marquées de $\mathcal{P}$. À tout moment de l'algorithme, soit une partie est coupée, ou marquée, et les parties marquées ne sont plus démarquées. Par conséquent il y a au plus $2n - 1$ appels à la procédure **RefineSelf**. Comme nous le verrons dans le lemme suivant la procédure **RefineSelf** s'exécute dans le pire des cas en $O(n^2)$ ce qui nous donne une complexité globale pour calculer $MU(S)$ en temps $O(n^3)$. □

LEMME 3.13. *Il est possible de calculer* **RefineSelf**$(\mathcal{P}, C)$ *en temps $O(n^2)$.*

DÉMONSTRATION. Dans un premier temps, montrons comment tester $R_C(x, y)$. Nous calculons, pour chaque élément x de C, une partition $\mathcal{H}(x, C) = \{P_x^1, \ldots, P_x^{k(x)}\}$ de $X \setminus C$. Il s'agit de la restriction de H_x à $X \setminus C$, *i.e.* $P_x^i = H_x^i \setminus C$. Il est facile de construire en temps $O(n)$ pour chaque élément de C. Alors nous avons $R_C(x, y)$ si et seulement si $\mathcal{H}(x, C)$ est la même partition que $\mathcal{H}(y, C)$.

Nous pouvons tester cela en temps $O(n)$, cependant procéder de la sorte et examiner tous les couples x, y nous conduirait à avoir une complexité globale en $O(n^3)$. Considérons plutôt $\mathcal{H}(x, C)$ comme un vecteur de b bits (où $b = n - |C| = O(n)$). Chercher les doublons

parmis ces vecteurs, peut se faire facilement à l'aide d'un tri par "bucket". En examinant leur premier bit, puis le second, etc ... Un parcours de tous les vecteurs (*i.e.* de tous les éléments de C), calcule des vecteurs deux à deux égaux, *i.e.* les classes d'équivalences de C. Il est ensuite facile de couper C et de mettre à jour $\mathcal{P}$ en temps $O(n^2)$. □

2.2. Umodules forts : calcul des umodules maximaux et test de primalité.

Comme pour les modules, nous pouvons utiliser le concept d'umodule fort. Un umodule U est dit fort, s'il n'existe aucun autre umodule U' dans $\mathscr{U}_H$ tels que $U \circledcirc U'$. Donc deux umodules forts sont soit disjoints, soit l'un est contenu dans l'autre. Nous rappelons que les umodules forts, tout comme les modules forts, forment une famille laminaire (*cf.* Schrijver (2003)).

THÉORÈME 3.14. *Il existe un algorithme qui admet une complexité en temps $O(n^5)$ pour calculer l'arbre d'inclusion des modules forts.*

DÉMONSTRATION. Considérons un umodule non-trivial U. Pour chaque couple d'éléments distincts $x, y \in U$ (il existe au moins deux éléments car U est non-trivial), U est contenu dans exactement un ensemble de $MU(\{x, y\})$. L'intersection de tous ces ensembles est exactement M. En effet, s'il y avait M' tel que $M \subsetneq M'$, alors il existerait $x \in M' \setminus M$. Pour $y \notin M$, $Mu(\{x, y\})$ contient un umodule U'' plus petit que M' mais qui contient M, d'où une contradiction.

L'algorithme est comme suit : pour chaque paire $\{x, y\}$, nous calculons $MU(\{x, y\})$ en temps $O(n^3)$ (*cf.* Théorème 3.12). Cela donne une famille d'au plus $O(n^3)$ umodules, à laquelle nous ajoutons les umodules triviaux $O(n)$.

Par la suite nous calculons l'intersection des umodules de la famille qui se chevauchent. Il est possible en un temps $O(n^4)$: pour chaque triplet (a, b, c) de chercher les umodules qui contiennent exactement deux des trois éléments, ils se chevauchent. Nous obtenons, par conséquent, tous les umodules forts. Ensuite, il faut ordonner umodules forts sous forme d'un arbre d'inclusion. □

L'algorithme fournit à la fois la liste des umodules forts et un test de primalité car un umodule non-trivial existe si et seulement si au moins umodule fort non trivial existe.

3. Deux Scénarios de Décompositions

Il est clair, comme dans le cas des modules, que la famille des umodules peut contenir beaucoup de membres, et par beaucoup, cela peut aller jusqu'à 2^n, quand n est la cardinalité de l'ensemble X. Nous allons maintenant nous concentrer sur des familles d'umodules qui admettent un codage compact de la famille. La première famille est la famille des umodules sur des relations homogènes de congruence locale égale à 2, la seconde famille est la famille des umodules auto-complémentée. En imposant ces restrictions sur chacune de familles,

cela nous permettra de les stocker en un espace polynomial. Nous obtenons $O(n^2)$ pour les relations homogènes de congruence locale 2 , et $O(n)$ pour les familles auto-complémentées.

3.1. Congruence locale et familles crossing.

Nous avons vu dans les définitions sur les relations homogènes que la congruence locale d'une relation homogène est le nombre maximum de classes associées à un élément (*cf.* définition 2.4 *p.* 24).

Les relations homogènes qui nous intéressent ici sont les relations homogènes de congruence locale égale à 2 (*i.e.* pour chaque x, H_x comporte au plus deux classes d'équivalences). En d'autres termes il s'agit de relations homogènes plus générales que les relations standard des graphes et des tournois (*cf.* X).

Quand la congruence locale d'une relation homogène est égale à 2, pour faire court, nous disons que H est $LC2$.

NOTATION 3.15 (Crossing $\dot{\circledcirc}$). Soit X un ensemble (fini), A et B deux sous-ensembles de X, A et B se croisent (cross en anglais) si : $A \circledcirc B$ et que $A \cup B \neq X$. On notera par $A \ \dot{\circledcirc} \ B$.

DÉFINITION 3.16 (Famille crossing). Soit X un ensemble fini, une famille $\mathscr{F}$ sur 2^X est dite crossing si pour tout sous-ensemble A et B appartenant à $\mathscr{F}$ tel que $A \ \dot{\circledcirc} \ B$ on ait :

$$A \cup B \in \mathscr{F} \text{ et } A \cap B \in \mathscr{F}$$

.

REMARQUE 3.17. Les familles crossing généralisent les familles Intersecting, qui elles même généralisent les familles ring (ou encore appelées lattice)

$$\text{Ring} \subseteq \text{Intersecting} \subseteq \text{Crossing}$$

PROPOSITION 3.18. *La famille des umodules sur les relations homogènes de congruence locale égale à 2 forme une famille* crossing.

DÉMONSTRATION. Sans formuler aucune hypothèse sur la relation homogène, nous savons par la proposition 3.2 que l'union de deux umodules qui se chevauchent est aussi un umodule.

Considérons maintenant deux umodules A et B tels que $A \circledcirc B$, par définition, $A \setminus B$ et $B \setminus A$ sont non vides. Soient $a \in A \setminus B$ et $b \in B \setminus A$. De plus pour que $A \cap B$ soit significatif, il doit y avoir au moins deux éléments (dans le cas contraire, il s'agit trivialement d'un umodule), notons les y et z. Et finalement notons x un élément de $X \setminus A \ \cup B$.

Par hypothèse nous avons

$$H(a|xb) \iff H(y|xb) \iff H(z|xb)$$

et

$$H(b|xa) \Longleftrightarrow H(y|xa) \Longleftrightarrow H(z|xa)$$

nous obtenons par la transitivité de H_z et x comme élément de "passage"

$$H(y|ab) \Longleftrightarrow H(z|ab).$$

Par conséquent $A \cap B$ est un umodule, et la famille des umodules sur les relations $LC2$ forme une famille crossing. □

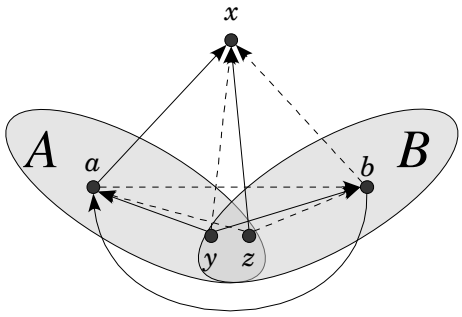


Figure 3.3. Illustration de la preuve de la proposition 3.18 : Les flèches sortantes représentent la relation homogène associée à chaque élément(ex : $H_y = \{\{a, b\}, \{x\}\} \ldots$)

Proposition 3.19. *La famille des umodules définie sur des relations homogènes de congruence locale supérieure ou égale à* 3 *ne forme plus une famille crossing.*

Démonstration. Considérons la relation homogène H de congruence locale égale à 3, définie comme suit :

$X = \{x, y, z, a, b\}$
- $H_a = \{\{b\}, \{x\}, \{\}\}$
- $H_b = \{\{x\}, \{a\}, \{\}\}$
- $H_y = \{\{a\}, \{b\}, \{x\}\}$
- $H_z = \{\{a, b\}, \{x\}, \{\}\}$

La relation partiellement définie ci dessus est bien de congruence locale au moins égale à 3 par y. De plus il est facile de voir que $\{a, y, z\}$ et $\{b, y, z\}$ sont des umodules, en effet

la partition non ordonnée quand l'on retire a est $\{\{b\},\{x\}\}$ (pour l'umodule $\{a,y,z\}$), et lorsque l'on retire b nous obtenons $\{\{a\},\{x\}\}$ (pour l'umodule $\{b,y,z\}$).

Par contre $\{y,z\}$ n'est pas un umodule car nous avons $\overline{H(y|ab)}$ et $H(z|ab)$. □

Remarque 3.20. La preuve de la proposition 3.18 s'appuie sur la propriété suivante, si nous avons :
$\overline{H(y|ab)} \wedge \overline{H(y|bc)}$ et la relation homogène est $LC2$ alors nous avons $H(y|ac)$, ce qui n'est pas nécessairement vrai lorsque la congruence locale est supérieure ou égale à 3.

Remarque 3.21. Remarquons que la famille umodules sur les relations homogènes $LC2$ est crossing et non intersectante car, il est nécessaire, pour avoir l'union de deux umodules qui se chevauchent, qu'il existe au moins un élément à l'extérieur.

Théorème 3.22 (Gabow (1993, 1995) ; Bernàth (2004)). *Il est possible de coder un famille crossing ayant pour ensemble support l'ensemble* $\{1,\ldots,n\}$ *en espace* $O(n^2)$.

Gabow (1993, 1995) a conçu une structure de donnée qui permet de stocker une famille Intersectante en espace $O(n^2)$. Il montre ensuite qu'une famille Crossing peut être codée par un nombre constant de familles Intersectantes (2 dans ce cas).

Plus récemment Bernàth (2004) a présenté une nouvelle structure pour coder ces familles.

3.2. Familles auto-complémentées et Familles Bipartitives.

Une conséquence du résultat précédent est que la famille des umodules des graphes non orientés (leur relation homogène standard en fait) et des tournois forment des familles crossing. Cependant ces deux structures ont des propriétés plus fines, qui vont nous permettre d'exhiber une structure en espace linéaire capable d'encoder la famille des umodules.

Définition 3.23 (Condition des 4 Points). Une relation homogène H, définie sur l'ensemble fini X, remplit la condition des 4 points si pour tout m,m',x,x' éléments de X les conditions suivantes sont satisfaites :

(1) $H(m|xx') \wedge H(m'|xx') \wedge H(x|mm') \Rightarrow H(x'|mm')$
(2) $\overline{H(m|xx')} \wedge \overline{H(m'|xx')} \wedge \overline{H(x|mm')} \Rightarrow \overline{H(x'|mm')}$

Proposition 3.24. *Les relations homogènes standard des graphes et des tournois satisfont la condition des* 4 *points.*

Définition 3.25 (Famille auto-complémentée). Soit X un ensemble fini, et $\mathscr{F}$ une famille sur 2^X. Une famille $\mathscr{F}$ est auto-complémentée si elle vérifie la propriété suivante : pour tout $A \subseteq X$ on a :

$$A \in \mathscr{F} \Longleftrightarrow \bar{A} \in \mathscr{F}$$

.

PROPOSITION 3.26. *Soit H une relation homogène définie sur l'ensemble X, qui satisfait la condition des 4 points, alors la famille des umodules $\mathscr{U}_H$ de H est auto-complémentée.*

DÉMONSTRATION. Dans un premier temps, remarquons que dans les cas des umodules triviaux, la proposition est trivialement vraie. En effet pour tout singleton, un umodule trivial, $\{x\}$, $X \setminus \{x\}$ est aussi un umodule, qui plus est trivial.

Considérons maintenant que U soit un umodule, mais que $X \setminus U$ ne le soit pas, *i.e.* il y x, x' deux éléments de $X \setminus U$, et m, m' deux éléments de U tels que $H(x|mm')$ et $\overline{H(x'|mm')}$.

Comme U est un umodule, soit m et m' distinguent en même temps x de x' (en clair $\overline{H(m|xx')}$ et $\overline{H(m'|xx')}$) ou ne distinguent pas du tout (*i.e.* $H(m|xx')$ et $H(m'|xx')$). Le premier cas est interdit par la première condition (1) de la définition 3.23, et le second cas est interdit par la seconde condition (2). □

La condition des 4 points est pour le moins pratique, car elle permet de réduire tout un umodule en un seul point, et par la suite d'appliquer le paradigme *"diviser pour régner"* afin de résoudre des problèmes d'optimisation.

Cependant, aussi longtemps que les umodules sont concernés, la propriété d'être auto-complémentée est suffisante pour décrire un théorème de décomposition sous forme d'arbre, comme nous le verrons dans la section suivante.

Cependant, remarquons que la réciproque à la proposition 3.26 n'est pas nécessairement vraie. La caractérisation, par un axiome local, des relations homogènes qui ont admette que la famille des umodules soit auto-complémentée, nous appairait être plus difficile.

3.3. Théorème de décomposition.

Nous présentons maintenant les propriétés bien connues de certaines familles de bipartitions. Les résultats suivants peuvent être retrouvés dans les travaux de Cunnigham et Edmonds (1980) sous la dénomination *"Decomposition Frame with the intersection and transitivity properties"*, théorie redécouverte dans les travaux de Montgolfier (2003) sous le nom de *"familles bipartitives"*, nom que nous conserverons dans la suite du texte, et enfin considérées sous le nom de *"unrooted set families"* dans les travaux de Hsu et McConnell (2003).

Nous appellerons $\{X_i^1, X_i^2\}$ une bipartition de X si $X_i^1 \cup X_i^2 = X$ et $X_i^1 \cap X_i^2 = \varnothing$.

DÉFINITION 3.27 (Chevauchement de Bipartitions). Deux bipartitions $\{X_i^1, X_i^2\}$ et $\{X_j^1, X_j^2\}$ se chevauchent si pour tout $a, b = 1, 2$ les 4 intersections $X_i^a \cap X_j^b$ sont non vides. Une bipartition est triviale si une des deux parties comporte un seul élément.

Soit $\mathscr{B} = \{\{X_1^1, X_1^2\}, \ldots, \{X_k^1, X_k^2\}\}$ une famille de k bipartitions de X. Les bipartitions fortes de $\mathscr{B}$ sont les bipartitions qui ne chevauchent aucune autre bipartition de $\mathscr{B}$. Par exemple les bipartitions triviales de $\mathscr{B}$ sont des bipartitions fortes.

PROPOSITION 3.28. *Si $\mathscr{B}$ contient toutes les bipartitions triviales de X, alors il existe un unique arbre $T(\mathscr{B})$ tel que :*

- *Comporte n feuilles, chaque feuille étant étiquetée par un élément de X.*
- *Chacune des arêtes e de $T(\mathscr{B})$ correspond à un bipartition forte de $\mathscr{B}$: Les étiquettes des feuilles de chacune de composantes de $T(\mathscr{B}) - e$ correspondent exactement à une des deux parties de la bipartition forte, la réciproque est également vraie.*

Soit N un noeud de $T(\mathscr{B})$ de degré k. Les étiquettes des feuilles des composantes connexes de $T - N$ forment un partition $X_1, \ldots, X_k$ de X. Pour $I \subseteq 1, \ldots, k$ avec $1 < |I| < k$, la bipartition $B(I)$ est $\{\cup_{i\in I} X_i, X \setminus \cup_{i\in I} X_i\}$.

DÉFINITION 3.29 (Familles Bipartitives Cunningham (1973) ; Montgolfier (2003)). Une famille de bipartitions est une *famille bipartitive* si elle contient toutes les bipartitions triviales et si, pour deux bipartitions qui se chevauchent, $\{X_i^1, X_i^2\}$ et $\{X_j^1, X_j^2\}$, les quatre bipartitions $\{X_i^a \cup X_j^b, X \setminus (X_i^a \cup X_j^b)$ (pour tout $a, b = 1, 2$) appartiennent à $\mathscr{B}$.

THÉORÈME 3.30. *Si $\mathscr{B}$ est une famille bipartitive, les noeuds de $T(\mathscr{B})$ peuvent être de types : Complet, circulaire ou enfin premiers. Les enfant des noeuds circulaires peuvent être ordonnés de telle manière que :*

1. *Si N est un noeud complet, for tout $I \subseteq \{1, \ldots, k\}$ tel que $1 < |I| < k$, $B(I) \in \mathscr{B}$.*
2. *Si N est un noeud circulaire, pour tout intervalle $I = [a, \ldots, b]$ de $\{1, \ldots, k\}$ tel que $1 < |b - a| < k$, $B(I) \in \mathscr{B}$.*
3. *Si N est un noeud premier, pour tout élément $I = \{a\}$ de $\{1, \ldots, k\}$ $B(I) \in \mathscr{B}$.*
4. *Il n'y a pas d'autres bipartition dans $\mathscr{B}$ que celles décrites ci dessus.*

Pour une famille bipartitive $\mathscr{B}$, l'arbre étiqueté $T(\mathscr{B})$ admet une taille en $O(n)$, et cet arbre représente la famille $\mathscr{B}$, alors que la famille peut avoir une taille (en nombres d'éléments) qui peut aller jusqu'à $2^{n-1} - 1$. Cette représentation compacte nous permet d'accomplir efficacement des opérations algorithmiques sur la famille $\mathscr{B}$.

Remarquons que toute famille auto-complémentée, peut être vue comme une famille de bipartitions.

PROPOSITION 3.31. *Les membres d'une famille auto-complémentée d'umodules forment une famille bipartitive.*

DÉMONSTRATION. Comme par hypothèse la famille des umodules $\mathscr{U}_H$ est auto-complémentée, nous pouvons associer une famille de bipartitions $Um'(H)$ qui comporte les "mêmes" éléments de la familles. Donc chaque partie d'une bipartition contenu dans $Um'(H)$ est un umodule. De plus si nous avons deux bipartitions $\{U, X \setminus U\}$ et $\{V, X \setminus V\}$ qui se chevauchent (tel que défini en 3.27). Alors $U \circledcirc V$ et en appliquant le résultat de la proposition

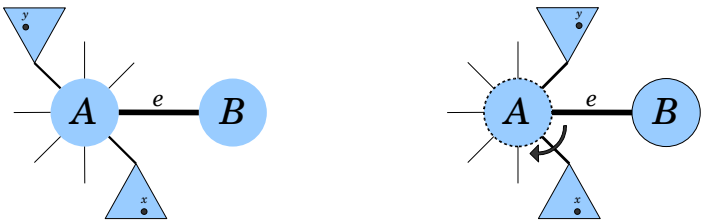
3.2 $U \cup V$ est un umodule. Nous en déduisons, comme la famille est auto-complémentée, que $\{U \cup V, X \setminus (U \cup V)\}$ appartient à $\mathscr{U}'(H)$. Nous procédons de manière similaire pour obtenir les trois bipartitions restantes. □

COROLLAIRE 3.32 (Théorème de Décomposition Umodulaire). *Il existe un unique arbre de décomposition de taille $O(n)$ qui encode tous les umodules possibles d'une relation homogène H auto-complémentée. Nous appelons naturellement cet arbre l'arbre de décomposition umodulaire.*

Remarquons simplement que contrairement à l'arbre de décomposition modulaire, l'arbre qui nous intéresse dans le cas présent est non enraciné.

3.3.1. *Algorithme de calcul de l'Arbre de Décomposition.* Soit H une relation homogène auto-complémentée, et soit $T_{\mathscr{U}}(H)$ son arbre de décomposition umodulaire, et enfin soit U un umodule fort non-trivial (s'il en existe). Examinons quelques unes des conséquences du théorème 3.30. Notons que deux umodules se chevauchent si et seulement si ils sont incidents à un même noeud de $T_{\mathscr{U}}(H)$. De plus comme H est auto-complémentée, l'union de deux umodules qui se chevauchent est un umodule (*cf.* Proposition 3.2), mais également leur intersection (en passant au complément). Un umodule fort U de H correspond à une arête e dans $T_{\mathscr{U}}(H)$ qui est incidente à deux noeuds A et B.

- Si, sans perte de généralité, A est étiqueté *premier*, alors pour tout $x, y \notin U$ tel que le plus petit ancêtre commun de x et y dans $T_{\mathscr{U}}(H)$ est A, alors U appartient à $MU(\{x, y\})$ (*cf.* Figure 3.4(a)).
- Si, sans perte de généralité, A est étiqueté comme un noeud *circulaire*, alors pour tout x appartenant au sous-arbre enraciné qui est le successeur de U selon l'ordre de A, et pour tout y appartenant au sous-arbre enraciné comme successeur de U selon l'ordre de A, alors U appartient à $MU(\{x, y\})$ (*cf.* Figure 3.4(b)).
- Si, sans perte de généralité, A est étiqueté *complet* alors , pour tout $x, y \notin U$ dont le plus petit ancêtre commun est A, l'intersection de toutes les parts de $MU(\{x, y\})$ contenant U est exactement U.



(a) A premier (b) A circulaire

Figure 3.4. Différents cas possibles

Le théorème 3.14 peut être utilisé pour calculer l'arbre d'inclusion des umodules forts. Après cette étape, typer les noeuds et ordonner leurs enfants selon la définition précédente est direct. Par conséquent :

THÉORÈME 3.33. *Il existe un algorithme qui calcule l'arbre de décomposition umodulaire, unique, pour une famille d'umodule auto-complémentée en temps* $O(n^5)$.

DÉMONSTRATION. Dans un premier temps nous calculons l'ordre d'inclusion des umodules forts en utilisant le théorème 3.14. Ensuite, nous typons les noeuds internes et ordonnons leurs enfant en fonction des définitions ci-dessus. □

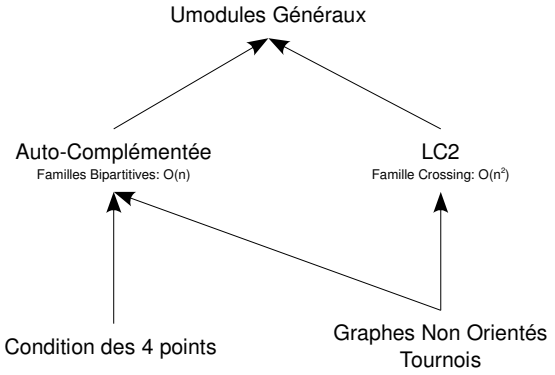


Figure 3.5. Différents types de famille d'umodules et leur codage respectifs

4. Un théorème puissant : Le Seidel-Switch

Les relations homogènes standard des graphes et des tournois sont de congruence locale égale à 2 et, de plus, la famille des umodules est auto-complémentée. Comme première conséquence de cette observation est que nous pouvons encoder la famille des umodules sur ces structures en utilisant au choix la représentation comme famille crossing, ou en utilisant l'arbre des familles bipartitives. De plus, les relations qui sont à la fois de congruence locale égale à 2 et auto-complémentées semblent présenter des propriétés structurelles plus fortes. En particulier, nous montrons comment transformation locale de la relation homogène, met en correspondance les umodules d'une relation avec les modules de la relation ainsi transformée. Cette opération fut introduite dans les travaux Seidel (1976) sur les graphes non-orientés. Cette transformation, et les questions de complexité qui y sont liées ont été étudiées par Colbourn et Corneil (1980) ; Kratochvíl et al. (1992), et les propriétés structurelles de cette transformation ont été étudiées par Hayward (1996) et Hertz (1999)

et plus récemment par Montgolfier et Rao (2005a,b). Cette opération est appelée *Seidel switch* dans les travaux de Hertz (1999), dans la suite du texte nous adopterons cette terminologie.

Nous généralisons cette transformation aux relations homogènes, en imposant cependant certaines restrictions par rapport à la définition originale. En effet, la différence réside dans le fait que l'élément utilisé pour effectuer la transformation n'intervient plus dans la relation ainsi obtenue.

4.1. Théorème du Seidel Switch.

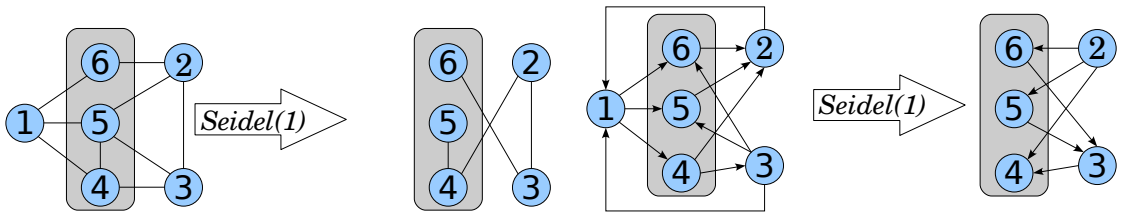
DÉFINITION 3.34 (Seidel Switch — Définition Originale). Soit $G = (V, E)$ un graphe non-orienté, et S un sous ensemble de sommets de V. Procéder au Seidel switch sur G avec S revient à supprimer toutes les arêtes présentes entre S et $\bar{S}$ et de rajouter toutes celles qui étaient absentes.

DÉFINITION 3.35 (Seidel Switch). Soit H une relation homogène de congruence locale 2 définie sur X. Soit s un élément de X. Le *Seidel Switch* en s transforme H en une relation homogène $H(s)$ sur $X \setminus \{s\}$ de manière suivante :

$$\forall x \in X \setminus \{s\} : \begin{cases} H(s)^1_x = (H^1_x \ \Delta H^j_s) \setminus \{s\} \\ \\ H(s)^2_x = (H^2_x \ \Delta H^j_s) \setminus \{s\} \end{cases}$$

avec j de telle sorte que $x \notin H^j_s$ et où $A\Delta B$ est la différence symétrique de A et B.

Quand nous appliquons cette définition aux graphes non orientés (*resp.* tournois), le Seidel switch échange simplement les arêtes et les non arêtes (*resp.* arcs entrants et arcs sortants) entre $N(x)$ et $\overline{N(x)}$ (*resp.* $N^+(x)$ et $N^-(x)$), voir figure 3.6.



(a) Sur un graphe non-orienté. (b) Sur un Tournoi.

Figure 3.6. Seidel Switch en action

THÉORÈME 3.36 (Théorème du *Seidel Switch*). *Soit H une relation homogène de congruence locale égale à 2 définie sur X telle que $\mathscr{U}_H$ soit auto-complémentée. Et soit s un élément quelconque de X, et $U \subseteq X$ un sous-ensemble contenant x. Alors U est un*

umodule de H si $M = X \setminus U$ est un module de $H(s)$ (la relation homogène sur $X \setminus \{s\}$ obtenue en effectuant un Seidel Switch en s).

Démonstration.
($\Rightarrow$) Soient $C = H_s^1 \cap M$ et $D = H_s^1 \cap M$. Comme H est de congruence locale égale à 2 C, D forme une bipartition de M. Soit $a \in U \setminus \{s\}$. Supposons que U soit un umodule de H. Alors pour tout y et z n'appartenant pas à U, nous avons $H(a|yz)$ si et seulement si $H(s|yz)$. En d'autres termes, C est inclus dans une des classe parmi H_a^1 et H_a^2 alors que D est inclus dans l'autre classe.

Comme $C \subseteq H_s^1$ et que $D \cap H_s^1 = \varnothing$, $C \cup D$ est inclus dans une parmi les deux classes $H(s)_a^i = H_a^i \Delta H_s^j$ ($i \in \{1, 2\}$ et j comme définit en définition 3.35).
Par conséquent, $M = C \cup D$ est un module de $H(s)$.
($\Leftarrow$) Réciproquement, si M est un module de $H(s)$, alors $C \cup D$ est alors inclus soit dans $H(s)_a^1$ soit dans $H(s)_a^2$. De plus, la définition de Seidel switch peut aussi être écrite sous la forme : $H_a^i = H(s)_a^i \Delta H_s^j$ pour i dans $\{1, 2\}$ et j tel que définit en définition 3.35. De plus, C est soit inclus dans H_a^1, soit dans H_a^2, alors que D est inclus dans l'autre classe. En d'autres termes, pour tout $a, b \in U$ et $y, z \notin U$, nous avons $H(a|yz)$ si et seulement si $H(b|yz)$ et par conséquent U est un umodule. □

Remarquons que l'arbre de décomposition modulaire de H peut être trivial alors que celui obtenu après Seidel Switch en s ne l'est peut-être pas.

4.2. Relations entre les umodules et les modules du Seidel switch.

Il est désormais connu que l'arbre de décomposition modulaire a beaucoup de propriétés structurelles bien étudiées Chein et al. (1981) ; Möhring et Radermacher (1984). Comme nous l'avons vu au chapitre précédent, il s'agit d'arbres enracinés, dont les feuilles sont en bijection avec les éléments de X. Un noeud de l'arbre de décomposition modulaire correspond à un module fort. Pour un noeud N soit $F_1, \ldots, F_k$ l'ensemble de ses k enfants dans l'arbre.

Quand la famille des umodules est bipartitive, comme c'est le cas pour le théorème 3.36, la famille des modules de toute relation obtenue après un Seidel switch sur H est une famille partitive (*cf.* Chein et al. (1981)), aussi connue sous le nom de *"famille enracinée"* dans les travaux de Hsu et McConnell (2003).

Le théorème 2.38 *p.* 39 décrit la structure des familles partitives. Les relations entre l'arbre de décomposition umodulaire de H et l'arbre de décomposition modulaire de $H(s)$ sont très étroites.

Proposition 3.37. *Soit H une relation de congruence locale égale à 2 définie sur X telle que $\mathscr{U}_H$ soit auto-complémentée. Soit s un élément de X. L'arbre de décomposition*

umodulaire $T_{\mathscr{U}}(H)$ et l'arbre de décomposition modulaire $\mathcal{T}(H(s))$ ont les propriétés suivantes :

(1) *Les deux arbres sont exactement les mêmes (noeuds et arêtes) à ce détail près que $\mathcal{T}(H(s))$ ne contient pas la feuille associée à s.*

(2) *Le noeud de $T_{\mathscr{U}}(H)$ qui est adjacent à la feuille contenant s est le noeud qui correspond à la racine dans $\mathcal{T}(H(s))$.*

(3) *Un noeud circulaire de $T_{\mathscr{U}}(H)$ correspond à un noeud linéaire de $\mathcal{T}(H(s))$; l'ordre sur les enfants reste inchangé. Les noeuds premiers et complets sont les mêmes dans chacun des arbres.*

DÉMONSTRATION. Il s'agit d'une conséquence du théorème 3.36. Chaque module de $H(s)$ donne une bipartition forte de $T_{\mathscr{U}}(H)$, et la réciproque est vraie. Pour un noeud N de l'arbre de décomposition modulaire, pour toute union $\cup_{i\in I}\mathscr{F}_i$ de l'ensemble des feuilles de ses enfants, il y a une bipartition $\{\cup_{i\in I}\mathscr{F}_i, X \setminus (\cup_{i\in I}\mathscr{F}_i)\} = B(I)$, en utilisant les notations définies ci-dessus. Pour chaque bipartition des umodules de H, la partie qui contient s est supprimée, et l'autre partie est placée dans la famille des modules de $H(s)$. □

Un résultat similaire, plus détaillé, peut être trouvé dans les travaux de Hsu et McConnell (2003). Ils établissent dans cet article les relations entre la propriété des 1 consécutifs et l'ordonnancement circulaire des matrices booléennes, mais les transformations sont les mêmes.

THÉORÈME 3.38. *L'arbre de décomposition umodulaire d'une famille auto-complémentée sur une relation homogène de congruence locale égale à 2 définie sur X, peut être calculé en temps $O(n^2)$ (où bien sûr $n = |X|$).*

DÉMONSTRATION. En utilisant le Seidel switch, à partir d'un élément quelconque de X, nous obtenons une relation homogène qui a la propriété du quotient modulaire (*cf.* définition 2.22 *p.* 32). De plus d'après le théorème 3.36 le complémentaire d'un module M de $H(s)$ est un umodule, et comme la relation est auto-complémentée M est un umodule dans H.

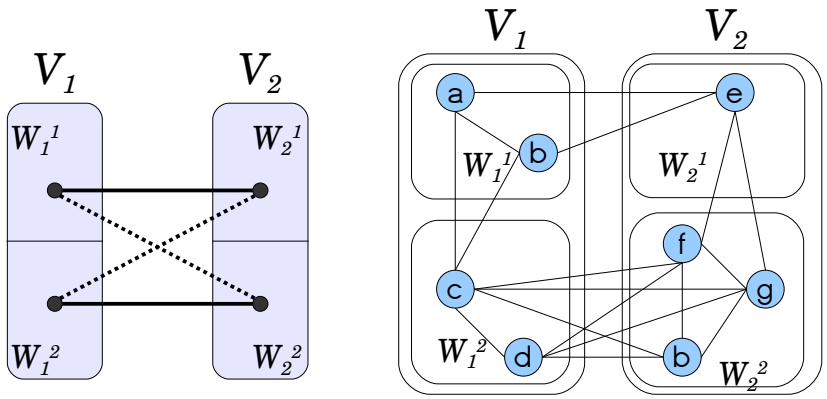
Il suffit donc de calculer le Seidel switch sur un élément s de X puis, nous pouvons utiliser l'algorithme 4 *p.* 53, qui calcule l'arbre de décomposition modulaire sur les relations homogènes qui possèdent la propriété du quotient modulaire. Sa complexité est $O(n^2)$ Il convient ensuite de rajouter s et de transformer les noeuds de l'arbre selon les dispositions de la proposition 3.37. Ce qui peut être facilement fait en temps linéaire. □

5. Décomposition umodulaire des graphes non-orientés

Nous allons maintenant appliquer cette décomposition, à une structure combinatoire bien connue que sont les graphes non-orientés, ou plus exactement la relation homogène standard des graphes non-orientés.

Sur les graphes non-orientés, cette décomposition était déjà connue, et a été présentée dans les travaux de Montgolfier et Rao (2005a,b). Nous présentons ici quelques unes des propriétés connues et nous établissons le lien avec les umodules.

Définition 3.39 (Bi-join). Un *bi-join* dans un graphe $G = (V, E)$ est une bipartition de l'ensemble des sommets V_1, V_2 telle que les arêtes entre V_1 et V_2 forment au plus deux graphes bipartis complets disjoints. En d'autres termes, pour tout $i \in \{1, 2\}$ nous pouvons bi-partitionner V_i en W_i^1 et W_i^2 de telle sorte que W_i^1 soit complètement adjacent à W_j^1 et que W_i^2 soit complètement adjacent à W_j^2 (où $j = i + 1 \ mod \ 2$), un illustration de bi-joins est donnée en figure 3.7.



(a) Vision Schématique.

(b) Exemple sur un graphe non-orienté

Figure 3.7. Décomposition en Bi-joins

Proposition 3.40. *Si $\{V_1, V_2\}$ est un bi-join d'un graphe alors V_1 et V_2 sont chacun des umodules*

Démonstration. Tout sommet de V_1 v distingue un sommet de W_2^1 vis à vis d'un sommet de W_2^2, autrement dit $\overline{H(v|w_1w_2)}$ (où $w_1 \in W_2^1$ et $w_2 \in W_2^2$). Cependant v ne peut pas distinguer deux sommets de W_2^1 ni de W_2^2. Par conséquent V_1 est un umodule. La preuve est identique pour V_2. □

Dans les travaux de Montgolfier et Rao (2005a,b) le Seidel switch est l'outil principalement utilisé pour obtenir les résultats structurels de cette décomposition.

PROPOSITION 3.41. *Soit $G = (V, E)$ un graphe non-orienté, et $\{V_1, V_2\}$ est un bi-join de G si et seulement si pour tout $v \in V_1$ (resp. V_2) V_2 (resp. V_1) est un module dans le graphe obtenu après qu'un Seidel switch soit effectué en v.*

DÉMONSTRATION. Immédiat, comme la relation standard d'un graphe est de congruence locale égale à 2 et que les bi-joins sont auto-complémentés, nous pouvons donc appliquer le résultat du théorème 3.36. □

COROLLAIRE 3.42. *La décomposition umodulaire d'un graphe équipée avec sa relation homogène standard est exactement la décomposition en bi-join.*

Parmi les conséquences présentées dans les travaux de Montgolfier et Rao (2005a,b), les bi-joins et donc la décomposition umodulaire ne possède pas de noeuds circulaires.

THÉORÈME 3.43 (Montgolfier et Rao (2005b)). *Il existe un unique arbre de décomposition T associé à un graphe non-orienté G. Tous les noeuds sont étiquetés* complet *ou* premier*. Il y a exactement deux types de noeuds complets :*

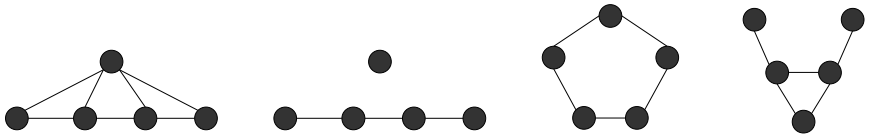
(1) *Les noeuds cliques K_n.*

(2) *Les noeuds bipartis complets $K_{n,m}$.*

5.1. Graphes totalement décomposables et Isomorphisme.

Nous montrons dans cette section que le problème de l'isomorphisme des graphes totalement décomposable avec la décomposition en bi-join est un problème polynomial. L'algorithme proposé est même linéaire. Cette classe de graphe a été étudiée par Montgolfier et Rao (2005a,b) sans que le problème de l'isomorphisme soit considéré.

THÉORÈME 3.44 (Montgolfier et Rao (2005a,b)). *Les graphes totalement décomposable pour la décomposition en bi-joins sont les graphes qui ne contiennent pas de C_5, de taureau, de gemme et de co-gemme, cf. la figure 3.8.*



(a) Gemme (b) Co-Gemme (c) C_5 (d) Taureau

Figure 3.8. Les sous-graphes induits minimaux interdits pour les graphes totalement décomposables par décomposition bi-join

REMARQUE 3.45. Nous ne reproduisons pas ici la preuve, nous remarquons simplement que sur ces graphes, procéder à un Seidel switch à partir de n'importe quel sommet engendre un P_4. Le principe de la preuve consiste à vérifier que ce sont les seuls graphes (minimaux) pour lesquels le phénomène se produit.

Ces graphes totalement décomposable par bi-join sont également les graphes qui sont construits par une séquence de jumeaux et anti-jumeaux. Par conséquent constituent une généralisation stricte des cographes.

La définition de jumeaux peut être retrouvée en définition 1.13 *p.* 9.

DÉFINITION 3.46 (Anti-Jumeaux Olariu (1988)). Soit $G = (V, E)$ un graphe non-orienté. Soient u et v deux sommets de G. u est v sont des anti-jumeaux si et seulement si :

(1) Vrais : $N(u) = V \setminus N(v)$.
(2) Faux : $N(u) = V \setminus (N(v) \cup \{u, v\})$

(a) Faux anti-jumeaux. (b) Vrais anti-jumeaux.

Figure 3.9. u et v deux sommets anti-jumeaux.

Pour un graphe complètement décomposable, par définition, il est clair que l'arbre de décomposition en bi-join ne comporte pas de noeuds premiers. De plus l'arbre de décomposition a une taille en $O(n)$ et permet de coder complètement le graphe (comme c'est le cas pour le co-arbre des cographes). Le problème de l'isomorphisme se réduit donc à tester l'isomorphisme d'arbre.

THÉORÈME 3.47 (Isomorphisme et Décomposition Bi-Join). *Soient G_1 et G_2 deux graphes complètement décomposables par la décomposition en bi-join. L'isomorphisme entre G_1 et G_2 peut être testé en temps linéaire.*

DÉMONSTRATION. Avec le théorème 3.36 et les résultats de Montgolfier et Rao (2005b), un arbre de décomposition umodulaire unique pour chacun des deux graphes peut être obtenu en temps $O(n + m)$.

Il suffit ensuite de tester l'isomorphisme d'arbres. Le problème de l'isomorphisme d'arbre est bien connu pour être linéaire, on pourra consulter les travaux de Aho et al. (1974).

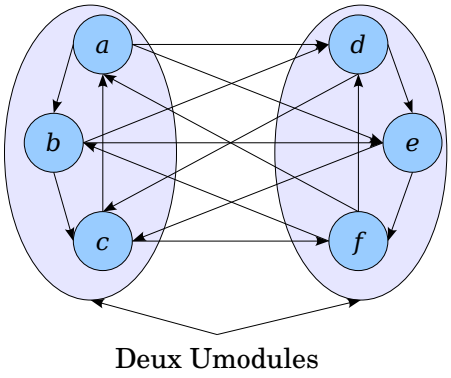
Remarquons simplement que les deux arbres obtenus ne sont pas enracinés et les noeuds internes sont étiquetés soit par des cliques, soit par des bipartis complets. Pour l'isomorphisme d'arbres il convient donc de trouver une racine potentielle pour chaque arbre. En prenant le centre de l'arbre comme racine, et comme il y au plus deux centres dans un arbre, il suffit de procéder à deux tests d'isomorphisme d'arbres. □

6. Une nouvelle décomposition des Tournois

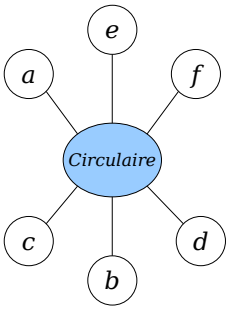
Nous avons étudié, dans la section précédente à quoi correspondait les umodules sur les graphes non orientés, et nous avons vu que cela nous a amené à une décomposition intéressante. De manière similaire, notre théorie peut s'appliquer aux tournois, et nous présentons une nouvelle décomposition pour les tournois : la décomposition umodulaire. En réalité, il s'agit de la décomposition umodulaire de la relation standard des tournois, qui rappelons le, est de congruence locale égale à 2. Nous verrons par la suite que la famille des umodules sur les tournois est auto-complémentée.

En ce qui concerne les tournois, cette décomposition est plus puissante que la décomposition modulaire (et même que la décomposition en coupe telle que définie par Cunningham (1982)). En effet tout module dans un tournois est également un umodule, mais la réciproque n'est pas vraie.

Considérons un tournoi M-premier (*resp.* U-Premier) s'il ne contient pas de module (*resp.* umodule) non-trivial. Nous pouvons donc décomposer des tournois M-Premiers avec la décomposition umodulaire (*cf.* figure 3.10).



(a) Un tournoi premier pour la décomposition modulaire, et décomposable pour la décomposition umodulaire.



(b) Arbre de décomposition Umodulaire non-enraciné.

Figure 3.10. Un exemple de tournoi indécomposable par la décomposition modulaire et son arbre de décomposition pour la décomposition Umodulaire.

Nous pouvons déduire de la proposition 3.37 des propriétés très intéressantes sur l'arbre de décomposition des tournois.

COROLLAIRE 3.48. *L'arbre de décomposition umodulaire d'un tournoi n'a pas de noeud complet. Et il y a un ordre circulaire des sommets du tournoi, tel que chaque umodule est un facteur (intervalle) de cet ordre circulaire.*

DÉMONSTRATION. La première observation vient du théorème 3.36 : après un Seidel switch à partir d'un sommet arbitraire s, nous obtenons un tournoi $H(s)$. Il est bien connu (*cf.* Montgolfier (2003)) que l'arbre de décomposition modulaire d'un tournoi ne comporte pas de noeud complet. Par la suite en appliquant la proposition 3.37 à partir d'un sommet s : l'arbre de décomposition umodulaire $T_{\mathscr{U}}(H)$ ne contient pas de noeud complet car l'arbre de décomposition modulaire $\mathcal{T}(H(s))$ ne contient pas non plus de noeud complet. Le second point est direct à partir du théorème 3.30. □

Ce résultat est déjà connu pour la décomposition modulaire (*cf.* Capelle (1997) ; Capelle et al. (2002) ; Montgolfier (2003)). Il existe une permutation (factorisante) des sommets, où chaque module est un facteur (intervalle). Il s'agit d'une permutation factorisante, déjà rencontrée au chapitre 2, définition 2.48 *p.* 49.

PROPOSITION 3.49. *La décomposition umodulaire d'un tournoi peut être calculée en temps $O(n^2)$.*

DÉMONSTRATION. Une fois encore, il convient d'utiliser le théorème 3.36, de procéder à un Seidel switch sur un sommet s, et par la suite de calculer l'arbre de décomposition modulaire du tournoi $H(s)$. Cela peut être fait en temps $O(n^2)$ en utilisant l'algorithme présenté par McConnell et Montgolfier (2005a,b). Ensuite la proposition 3.37 nous dit comment transformer l'arbre de décomposition modulaire $\mathcal{T}(H(s))$ en arbre de décomposition umodulaire $T_{\mathscr{U}}(H)$. □

Étant donné un schéma de décomposition, il est souvent intéressant de considérer les graphes qui sont totalement décomposables vis à vis de la décomposition en question. On pourra se rapporter à la définition 1.23 *p.* 12. En général cela amène d'intéressantes classes de graphes, tels que les cographes (*cf.* définition 1.25 *p.* 13) qui sont les graphes totalement décomposables pour la décomposition modulaire ou encore les graphes distances héréditaires qui sont les graphes complètement décomposables par la décomposition en coupes.

Nous investiguons ici quelles sont les classes des tournois totalement décomposables par la décomposition umodulaire.

6.1. Tournois localement transitifs.

Dans cette section nous nous concentrons sur l'étude des tournois complètement décomposables pour la décomposition umodulaire. Nous montrons les relations structurelles entre la classe de cographes (voir Brandstädt et al. (1999)) et les classe des tournois localement transitifs, aussi connus comme les *round* tournois qui sont une classe des digraphes semi-complets (*cf.* Bang-Jensen et Gutin (2000) pour plus de détails).

PROPOSITION 3.50. *Soit $T = (V, A)$ un tournoi, nous avons les propriétés suivantes :*
- *Si T est M-premier, alors il contient un circuit de taille 3 comme sous-graphe induit (cf. 3.11(a)).*
- *T est totalement décomposable vis à vis de la décomposition modulaire si et seulement si il ne contient pas de circuit à 3 sommets (i.e. un tournoi transitif – figure 3.11(b)).*

(a) Circuit à 3 sommets

(b) Tournoi Transitif

Figure 3.11. Tournoi transitifs et Ciruit

6.2. Théorème de Caractérisation.

Nous avons :

THÉORÈME 3.51 (Caractérisation des Tournois Totalement Décomposables). *Soit $T = (V, A)$ un tournoi. Si T est un tournoi U-Premier alors T contient un diamant (un des graphes présentés en figure 3.12).*

T est totalement décomposable vis à vis de la décomposition umodulaire si et seulement si T ne contient pas de diamants comme sous-graphes induits.

DÉMONSTRATION. Grâce au théorème 3.36, un tournoi $T = (V, A)$ est U-Premier si et seulement si pour tout sommet v un Seidel switch en v donne un tournoi M-Premier. Et

grâce à la proposition 3.50, il suffit de regarder tous les tournois à 4 sommets et de voir lesquels, après un Seidel switch sur un sommet, nous donnent un circuit avec 3 sommets. Laborieux, mais pas difficile. □

(a) C_3 Dominant. (b) C_3 Dominé.

Figure 3.12. Sous-graphes Interdits pour les tournois totalement décomposables vis à vis de la décomposition umodulaire.

Toutefois, il est possible de donner une autre caractérisation :

DÉFINITION 3.52 (Tournoi Localement Transitif). Un tournoi $T = (V, A)$ est un tournoi *Localement Transitif* si pour tout sommet $v \in V$, $T[N^+(v)]$ et $T[N^-(v)]$ sont des tournois transitifs.

PROPOSITION 3.53. *Un tournoi $T = (V, A)$ est sans diamant si et seulement si il est localement transitif.*

DÉMONSTRATION. Immédiat. □

6.3. Algorithme de Reconnaissance.

Il est fréquent, dans l'étude des classes de graphes, de s'intéresser au problème de la reconnaissance des membres de cette classe. Nous montrerons que le problème de la reconnaissance des graphes totalement décomposable par la décomposition umodulaire est polynomial, et nous fournirons un algorithme linéaire certifiant pour reconnaître cette classe.

Grâce au théorème 3.51 il est facile de tester en un temps $O(n^4)$ si, étant donné un tournoi, il appartient à la classe. En effet il suffit de considérer tous les sous-tournois induit de taille 4 et voir si'l y a au moins un diamant. Une manière plus habile de procéder consiste à utiliser le résultat de la proposition 3.53, et regarder pour chaque sommet v de T si ses deux voisinages sont tous les deux des tournois transitifs. Ce qui nous donne un algorithme ayant une complexité en $O(n^3)$. Il est cependant possible de faire mieux, *i.e.* un algorithme

linéaire, en ne considérant qu'un seul sommet du tournoi et en utilisant d'autres propriétés inhérentes à cette famille.

Un autre algorithme linéaire pour reconnaître cette classe a été présenté par Clarou (1996). Nous présentons ici un algorithme différent de celui de Clarou basé sur le concept de permutation factorisante. Qui plus est, notre algorithme est certifiant, à savoir si le tournoi T donné en entrée n'est pas un tournoi totalement décomposable, il exhibe un diamant, et si par contre il est totalement décomposable, pour s'en assurer, il fournit l'arbre de décomposition umodulaire. Dans cet arbre il convient simplement de vérifier que n'appairassent aucun noeud premier. Dans les deux cas, le certificat a une taille soit linéaire pour l'arbre soit constante pour le sous-graphe induit. Par conséquent notre algorithme répond bien aux critères d'algorithme certifiant évoqués dans les travaux de Kratsch et al. (2003, 2006).

PROPOSITION 3.54. *Soit $T = (V, A)$ un tournoi, et soit v un sommet arbitraire de $V(T)$. T est localement transitif si et seulement si les conditions suivantes sont vérifiées :*

(1) *$T[N^+(v)]$ et $T[N^-(v)]$ sont des tournois transitifs.*

(2) *Si un sommet $a \in N^+(v)$ a voisin sortant $b \in N^-(v)$ et un voisin entrant $c \in N^-(v)$ alors $(b, c) \in A$.*

(3) *Si un sommet $a \in N^-(v)$ a un voisin sortant $b \in N^+(v)$ et un voisin entrant $c \in N^+(v)$ alors $(b, c) \in A$.*

DÉMONSTRATION.
($\Rightarrow$) Supposons que T soit un tournoi localement transitif, par définition même, (**1**) est nécessairement vrai. Si maintenant (**2**) n'est pas vrai pour des sommets a, b, c, *i.e.* s'il y a un (c, b) au lieu d'un arc (b, c), alors $\{a, b, c, v\}$ forme un des sous-graphe induit interdit (*cf.* figure 3.12). Il en est de même pour le point (**3**).
($\Leftarrow$) Réciproquement supposons que les trois conditions soient satisfaites. Nous devons montrer que pour chaque sommet, les tournois formés par ses deux voisinages sont des tournois transitifs. Et ensuite la proposition 3.53 nous dit qu'il est totalement décomposable.

Pour v c'est vrai grâce au point (**1**). Soit t un sommet de $N^+(v)$. Si $T[N^+(t)]$ n'est pas transitif, alors il contient un circuit (x, u, w) (avec comme arcs, sans perte de généralité (x, u), (u, w) et (w, x)). Comme $T[N^+(v)]$ et $T[N^-(v)]$ sont tous les deux transitifs, cela signifie que le circuit chevauche ces deux tournois. Supposons, sans perte de généralité que $x \in N^+(v)$ et $wN^-(v)$. Alors (**3**) n'est pas vérifié, en effet, si l'on prend $a = w$ et $b = x$ et $c = t$.

Si nous supposons que $T[N^-(t)]$ pas transitif, il contient un circuit (x, u, w) avec un arc (x, w), $x \in N^+(v)$ et $w \in N^-(v)$. (**3**) est une fois de plus violée : en prenant pour a, b et c : $a = w$, $b = t$ et $c = x$,.

Maintenant soit t un sommet de $N^-(v)$. Si $T[N^+(t)]$ n'est pas transitif, cela signifie qu'il contient un circuit (x, u, w) avec un arc (x, w), $x \in N^+(v)$ et $w \in N^-(v)$. Par conséquent la condition (**2**) est violée, en prenant $a = x$, $b = w$ et $c = t$. Et si $T[N^-(t)]$ n'est pas transitif, alors il contient un circuit (x, u, w) avec un arc (x, w) avec $x \in N^-(v)$ et $w \in N^+(v)$. Par conséquent la condition (**2**) est violée, en prenant $a = x$, $b = t$ et $c = w$. □

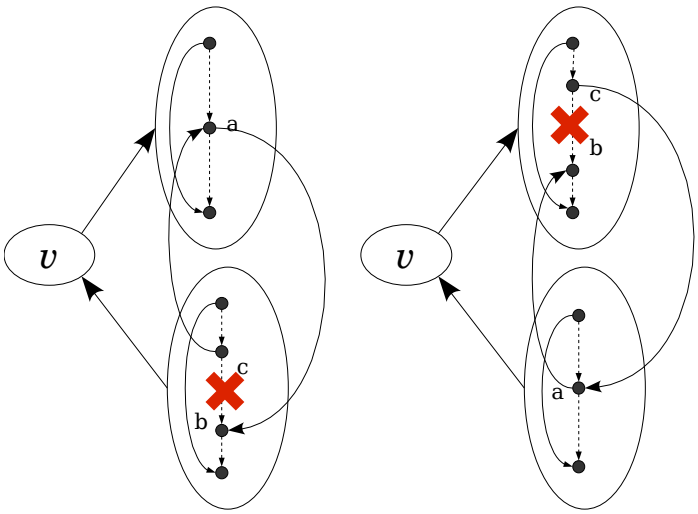


Figure 3.13. Configuration Interdites telles que décrites par la proposition 3.54.

THÉORÈME 3.55. *Il existe un algorithme polynomial et certifiant pour reconnaître la classe des tournois localement transitifs. Sa complexité est en temps* $O(n^2)$.

DÉMONSTRATION. La condition (**1**) de la proposition 3.54 peut être testée en temps $O(n^2)$. Nous numérotons $a_0, \ldots, a_k$ les sommets de $N^+(v)$ de manière croissante selon l'ordre transitif (*i.e.* l'ordre de l'extension linéaire) de $T[N^+(v)]$, et $b_0, \ldots, b_l$ les sommets de $N^-(v)$ de manière croissante.

Si a_i satisfait la condition (**2**), alors son voisinage sortant contient $b_0, \ldots, b_{f(i)}$ et son voisinage entrant $b_{f(i)+1}, \ldots, b_l$. Cela peut être testé en temps $O(n)$. Un test similaire en $O(n)$ est effectué pour chaque b_i et b_j, ce qui nous amène naturellement à un algorithme qui s'exécute en temps $O(n^2)$. □

L'algorithme 6 présente une implémentation certifiante de cette preuve.

Entrées : Un Tournoi $T = (V, A)$
Sorties : **Oui** : Un permutation factorisante circulaire σ de T.
Non : Une obstruction.

début
  Prendre un sommet $x \in V$
  A $\leftarrow N^+(x)$
  B $\leftarrow N^-(x)$
  si T[**A**] *n'est pas un tournoi transitif* **alors**
    Échec : le certificat est *Un 3-circuit est contenu dans A dominé par x*
  Ordonner $a_1, \ldots, a_k$ les sommets de A selon l'ordre total (a_1 est la source et a_k le puits)
  si T[**B**] *n'est pas un tournoi transitif* **alors**
    Échec : le certificat est *Un 3-circuit est contenu dans B anti-dominaté par x*
  Ordonner $b_1, \ldots, b_l$ les sommets de B selon l'ordre total (b_1 est la source et b_l le puits)
  pour $i \leftarrow 1$ **to** k **faire**
    $j \leftarrow 1$
    tant que $j \leq l$ *et* $b_j \in N^+(a_i)$ **faire**
      j++
    tant que $j \leq l$ *et* $b_j \in N^-(a_i)$ **faire**
      j++
    si $j \neq l + 1$ **alors**
      Échec : le certificat est $\{x, a_i, b_{j-1}, b_j\}$ *; b_{j-1} est la source*
      `// bien sûr alors` $b_j \in N^+(a_i)$ `mais` $b_{j-1} \in N^-(a_i)$ `et` $b_j \in N^+(b_{j-1})$
  pour $j \leftarrow 1$ **to** l **faire**
    $i \leftarrow 1$
    tant que $i \leq k$ *et* $a_i \in N^+(b_j)$ **faire**
      i++
    tant que $i \leq k$ *et* $a_i \in N^-(b_j)$ **faire**
      i++
    si $i \neq k + 1$ **alors**
      Échec : le certificat est $\{x, b_j, a_{i-1}, a_i\}$ *; un diamant où a_i est le puits*
      `// Bin sûr alors` $a_i \in N^+(b_j)$ `mais` $a_{i-1} \in N^-(b_j)$ `et` $a_i \in N^+(a_{i-1})$
  Choisir un sommet x quelconque
  Procéder à un Seidel switch en x
  $\sigma \leftarrow Seidel(x) \cup x$
  retourner $\sigma(V)$ *une permutation factorisante circulaire*
fin

Algorithme 6 : Algorithme de reconnaissance certifiant d'un tournoi totalement décomposable par la décomposition umodulaire. Le certificat renvoyé en cas d'échec est un diamant. En cas de succès le certificat est une permutation factorisante circulaire.

6.4. Abre de décomposition umodulaire d'un tournoi localement transitif.

THÉORÈME 3.56 (Umodules d'un tournoi localement transitif). *Soit $T = (V, A)$ un tournoi localement transitif. L'arbre de décomposition umodulaire de T comporte un seul noeud. De plus ce noeud est de type circulaire.*

DÉMONSTRATION. Grâce au théorème 3.36, pour tout Seidel switch en un sommet x d'un tournoi complètement décomposable par la décomposition umodulaire le tournoi obtenu $H(x)$ est totalement décomposable par la décomposition modulaire. La proposition 3.50 nous dit que seul les tournois transitifs sont totalement décomposable avec la décomposition modulaire, par conséquent $H(x)$ est transitif et son arbre de décomposition est constitué d'un seul noeud de type linéaire. Et donc selon la proposition 3.37, l'arbre de décomposition umodulaire de T a un seul noeud de type circulaire. □

Dans le cas des cographes, il est bien connu qu'il est suffisant pour coder le graphe de conserver son arbre de décomposition modulaire (son co-arbre). Malheureusement ce n'est pas le cas pour les tournois totalement décomposables par la décomposition umodulaire. En effet le fait qu'il y ait qu'un seul noeud de type circulaire (pas complètement caractérisé), ne permet pas de coder l'adjacence entre les sommets.

L'ordre circulaire des sommets autour de noeud circulaire unique est appelée une *permutation factorisante circulaire*, comme tout umodule de T est un intervalle de cette permutation, et la réciproque est également valide, par définition même d'un noeud circulaire.

Plus de résultats sur les tournois localement transitifs sont également connus.

6.5. Structure circulaire des tournois localement transitifs.

Une structure circulaire des tournois localement transitifs est connue dans les travaux de Lopez et Rauzy (1992a,b), nous les rappelons ici :

DÉFINITION 3.57. Un tournoi $T = (V, A)$ est un *circuit complet* si les sommets peuvent être numérotés de de 0 à $2k$ (k un entier positif) et si pour tout sommet numéroté i, son voisinage sortant est constitué des sommets numérotés de $i+1$ à $i+k$ inclus (le tout modulo $2k + 1$).

Un exemple de circuit complet est donné en figure 3.14.

THÉORÈME 3.58 (Lopez et Rauzy (1992a,b)). *Soit $T = (V, A)$ un tournoi localement transitif. V peut être partitionné en $V_0, \ldots, V_{2k}$, avec k un entier positif et :*

1. *Pour tout i, $0 \leqslant i \leqslant 2k$, $T[V_i]$ est un tournoi transitif.*
2. *Pour tout $x \in V_i$ et $y \in V_j$, sil existe $a \leqslant k$ tel que $i = a + j$ modulo $2k + 1$ alors (x, y) est un arc de T, dans le cas contraire (y, x) est un arc.*

Remarquons que tous les ensembles V_i constituent des modules du tournoi, de plus ces modules sont maximaux vis à vis de l'inclusion (*i.e.* ils sont forts, et le seul module à contenir V_i est V tout entier).

COROLLAIRE 3.59. *Un module non trivial M d'un tournoi localement transitif induit un tournoi transitif.*

L'ordre circulaire des $2k+1$ modules forts, *i.e.* la partition circulaire $V_0, \ldots, V_{2k}$ telle que définie dans le théorème 3.58, est désormais appelée *ordonnancement circulaire.*

Nous avons déjà rencontré une autre "structure circulaire" : la permutation factorisante circulaire des n sommets du tournoi. Il s'avère que ces deux "structures" ne sont pas isomorphes, cependant :

Soit $T = ([0, \ldots, 2k], A)$ l'unique circuit complet (à isomorphisme près) avec $2k+1$ sommets. Alors soit τ la bijection :

$$\tau(i) = ki \text{ modulo } 2k+1$$

Nous considérons par la suite la séquence τ comme un liste circulaire nous la noterons τ'.

PROPOSITION 3.60. *Les intervalles de τ' sont exactement les umodules de T.*

DÉMONSTRATION. Grâce à la propriété de clôture par union d'umodules se chevauchant (*cf.* proposition 3.2) il nous suffit simplement de vérifier que les umodules à deux sommets sont exactement les paires de la forme $\{i, i+k\}$ (le tout modulo $2k+1$). Ce qui est facile à vérifier. □

Cette proposition peut être généralisée aux tournois localement transitifs de la manière suivante :

PROPOSITION 3.61. *Soit $T = (V, A)$ un tournoi localement transitif et $V_0, \ldots, V_{2k}$ son ordonnancement circulaire. Chacun des ensembles V_i induit un tournoi transitif (i.e. ses sommets forment une chaîne de la forme suivante : $v_i^1, \ldots, v_i^{f(i)}$).*

Soit σ une permutation factorisante circulaire telle que

1. *Chaque V_i est un facteur (intervalle) de σ.*
2. *Les ensembles V_i suivent consécutivement τ', i.e. V_0 ensuite V_k ensuite V_{2k} puis V_{3k}... (les indices sont pris modulo $2k+1$).*
3. *À l'intérieur de chaque V_i l'ordre des sommets est l'inverse de l'ordre de la chaîne : $v_i^{f(i)}, \ldots, v_i^1$.*

Les umodules de T sont exactement les intervalles de σ. De plus σ est l'unique permutation factorisante circulaire de T.

La figure 3.15 donne un exemple de la relation qu'il y a entre la permutation factorisante circulaire et l'ordonnancement circulaire.

Cette proposition permet de construire l'ordonnancement circulaire, étant donné la permutation factorisante circulaire calculée par l'algorithme 6.

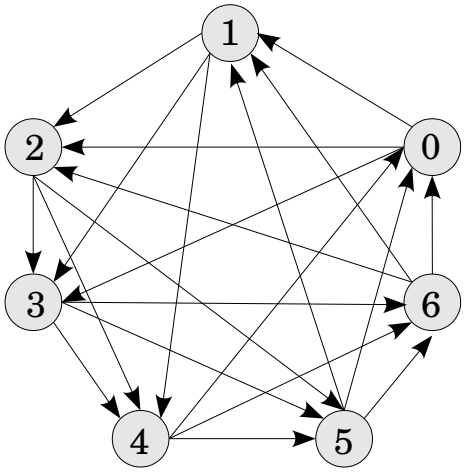


Figure 3.14. Un circuit complet à 7 sommets (avec $k = 3$).

Une première étape consiste à identifier les $2k + 1$ tournois induits. Deux sommets u et v sont *jumeaux* si $N^+(u) \setminus \{v\} = N^+(v) \setminus \{u\}$. Ce sont des *jumeaux consécutifs* pour une permutation factorisante circulaire s'ils se suivent de manière consécutive dans la permutation. Soit R la fermeture transitive de la relation être jumeaux consécutifs.

PROPOSITION 3.62. *Les classes d'équivalence de R sont exactement les tournois transitifs induits $V_0, \ldots, V_{2k}$ de l'ordonnancement circulaire d'un tournoi localement transitif.*

DÉMONSTRATION. Il n'est pas difficile de voir, dans un tournoi, que deux sommets jumeaux forment un module à deux sommets, et que les classes d'équivalences de la fermeture transitive de R sont par conséquent des modules.

Il suffit simplement d'appliquer le corollaire 3.59. Il faut maintenant vérifier que chaque classe M de R est un module *maximal* : s'il ne l'est pas, il existe x tel que $M \cup \{x\}$ est un module, mais donc, soit x et M sont des puits, soit x et M sont des sources, soit x et M sont jumeaux, donc contradiction. □

Nous pouvons donc donner un autre algorithme quadratique que celui donné par Clarou (1996)

THÉORÈME 3.63. *L'ordonnancement circulaire d'un tournoi peut être calculé en temps $O(n^2)$.*

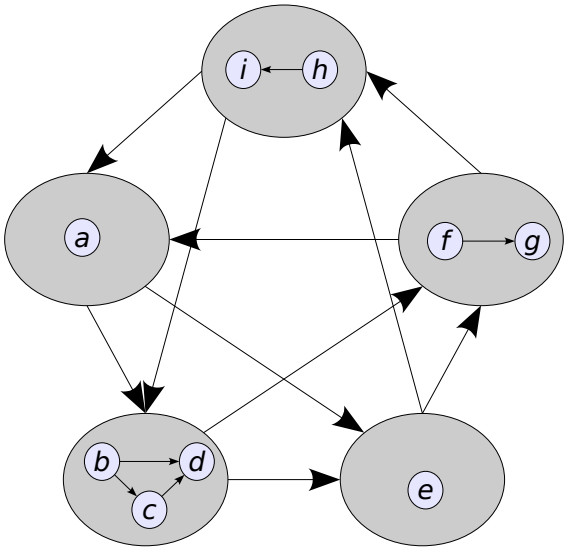


Figure 3.15. Un exemple de tournoi localement transitif avec 9 sommets : Le circuit complet à 5 sommets où chaque sommet est substitué par un tournoi transitif. Un ordonnancement circulaire est $(\{a\},\{b,c,d\},\{e\},\{f,g\}\{h,i\})$ alors qu'une permutation factorisante est (a,e,i,h,d,c,b,g,f).

DÉMONSTRATION. Dans un premier temps nous utilisons l'algorithme 6 pour calculer un permutation factorisante circulaire. Ensuite la relation R de la proposition 3.62 peut être calculée en vérifiant si les n paires de sommets sont jumeaux ou pas. Ensuite les ensembles $V_0,\ldots,V_{2k}$ sont réordonnés en utilisant l'inverse de τ tel que défini dans la proposition 3.61. □

6.6. Stockage efficace des tournois localement transitif.

DÉFINITION 3.64. Une composition de n est une liste de k termes entiers tels que la somme des termes est égale à n. La composition est *impaire* si k est impair.

Une *composition impaire* de n est une liste circulaire avec $2k+1$ termes entiers telle que la somme est égale à n. Remarquons qu'une liste circulaire admet un sens de lecture : en effet $(1,2,3)$ n'est pas équivalent à $(3,2,1)$ mais est identique à $(2,3,1)$.

Il a été montré dans les travaux de Brouwer (1980) une formule d'énumération des tournois localement transitifs. Pour ce faire il a établi une bijection entre les tournois et *"Shift registers where the complement of the bit shifted out of the last position is shifted into the first position"*. Que l'on peut reformuler de la manière suivante :

THÉORÈME 3.65 (Brouwer (1980)). *Il existe une bijection entre les tournois localement transitifs à n sommets et les compositions circulaires impaires de n éléments.*

Brouwer a donné les premiers termes de la séquence, à savoir le nombre de tournois totalement décomposable sur n sommets. Cette séquence est référencé dans l'encyclopédie des séquence d'entiers Sloane comme étant la séquence A000016 [1] :

1, 1, 1, 2, 2, 4, 6, 10, 16, 30, 52, 94, 172, 316, 586, 1096, 2048, 3856, 7286, 13798, 26216, 49940, 95326, 182362, 349536, 671092, 1290556, 2485534, 4793492, 9256396

Par ailleurs il a également donné la formule exacte :

$$\sum_{d|n} \frac{2^{d-1}}{d}\ odd(\frac{d}{n})\ .\ \sum_{e|\frac{n}{d}} \frac{\mu(e)}{e}$$

Où μ est la fonction de Möbius (*cf.* définition A.5 *p.* 133) et $odd(x)$ vaut 1 si x est impair, 0 sinon.

Remarquons que le nombre de tournois totalement décomposable vis à vis de la décomposition umodulaire est réellement plus grand que le nombre de tournois totalement décomposable avec la décomposition modulaire. En réalité pour la décomposition modulaire, et pour un entier n fixé, il n'existe qu'un seul tournoi totalement décomposable : il s'agit du tournoi transitif.

THÉORÈME 3.66. *Un tournoi localement transitif non étiqueté (resp. étiqueté) à n sommets peut être stocké en $O(n)$ (resp. $O(n\ log\ n)$) bits.*

DÉMONSTRATION. Si le tournoi est non étiqueté, il suffit simplement de stocker la composition circulaire impaire d'entiers correspondante. Cela peut être fait en utilisant une méthode standard de codage des compositions, à savoir : un vecteur de $n-1$ bits. Si le k-ième terme de la composition est x, il est codé par $x-1$ "1" suivi d'un "0". Le dernier bit est toujours un "0" et peut donc être supprimé.

Par exemple la composition $(2,3,1,1,3)$ de 10 est codé comme :
101100011. Il s'agit simplement d'un codage unaire (un peu modifié) des nombres de la composition. Il s'agit d'une technique classique de codage des compositions (*cf.* (Andrews, 1976, "Chapter 4: Compositions and Simon Newcomb's problem")).

Si maintenant, le tournoi est étiqueté, il est nécessaire de stocker la permutation des sommets, cela peut être fait en utilisant $O(n\ log\ n)$ bits. □

6.7. Plus Petit Ensemble de Sommets Retour.

[1] `http://www.research.att.com/~njas/sequences/A000016`

DÉFINITION 3.67 (Ensemble de sommets retour). Un ensemble de sommets retour dans un graphe orienté $G = (V, A)$ est un sous-ensemble de sommets X de V, tel que tous les éléments de V' appartiennent à au moins un circuit.

Le problème du *Plus Petit Ensemble de Sommets Retour* consiste à trouver un ensemble retour de taille minimum. Ce problème est rencontré dans la littérature sous le nom de *Minimum Feedback Vertex Set*.

Ce problème est connu pour être **NP**-Difficile sur les graphes orientés en général (Garey et Johnson, 1979, GT7), et demeure **NP**-Difficile sur les tournois Speckenmeyer (1989). Nous montrons dans cette section que le problème devient polynomial lorsqu'on le considère sur des tournois totalement décomposable par la décomposition umodulaire.

Une autre façon de considérer ce problème est de trouver un ensemble de sommets de taille (*i.e.* cardinalité) minimum de telle sorte que le graphe restant soit sans circuit. En conséquence de quoi, lorsque l'on considère les tournois, cela revient à chercher le plus grand (toujours en terme de cardinalité) sous-tournoi induit transitif.

Soit $\delta(T) = min\{\delta^+(T), \delta^-(T)\}$. Où $\delta^+(T)$ est le plus petit degré sortant de T et $\delta^-(T)$ le plus petit degré entrant de T.

LEMME 3.68. *Soit $T = (V, A)$ un tournoi localement transitif. Et soit x un sommet de T tel que soit $|N^-(x)| = \delta(T)$ soit $|N^+(x)| = \delta(T)$.*

Si $|N^-(x)| = \delta(T)$ alors $N^-(x)$ est le plus petit ensemble de sommets retour. Sinon c'est $N^+(x)$ qui est le plus petit ensemble de sommets retour.

DÉMONSTRATION. Supposons que $|N^-(x)| = \delta(T)$ (dans le cas contraire il s'agit du même raisonnement en passant au complémentaire). Comme T est localement transitif $T[\{x\} \cup N^+(x)]$ est un tournoi transitif, par conséquent $N^-(x)$ répond bien à la définition 3.67.

Réciproquement supposons que T contienne un ensemble retour F de taille strictement inférieure à $|N^-(x)|$. Comme nous avons pris, par hypothèse, le sommet avec le plus petit degré entrant, pour tous les sommets y, $T[V \setminus F]$ à la fois des sommets de $N^+(y)$ et $N^-(y)$. Par conséquent cela ne peut pas être un tournoi transitif. □

Une conséquence directe de ce lemme est

THÉORÈME 3.69. *Le plus petit ensemble de sommets retour d'un tournoi totalement décomposable par la décomposition umodulaire peut être trouvé en temps $O(n^2)$.*

L'algorithme consiste à trouver le sommet de plus petit degré et de supprimer le voisinage. Difficile de faire plus simple.

6.8. Isomorphisme.

Le problème de l'isomorphisme est un problème fondamental tant du point de vue de la complexité que du point de vue de la théorie des graphes.

À l'heure actuelle il n'existe pas de condition nécessaire et suffisante "efficace" pour déterminer si deux graphes sont isomorphes. Le statut de ce problème dans l'univers de la complexité est encore incertain, nous savons qu'il est dans **NP**, mais sans plus de précisions.

Bien que le problème soit polynomial pour certaines classes de graphes comme les arbres (*cf.* Aho et al. (1974)), les graphes planaires (*cf.* Hopcroft et Tarjan (1971, 1973) ; Hopcroft et Wong (1974)), ou encore les cographes, les graphes distance héréditaires, les graphes totalement décomposables par bi-joins (*cf.* section 5.1).

Quant aux tournois, le statut de l'isomorphisme semble toujours ouvert (*cf.* Babai et Luks (1983) ; Arvind et al. (2006)). Dans ses travaux Clarou (1996) a présenté un algorithme pour tester l'isomorphisme de deux tournois localement transitifs.

Il est facile de voir que, une fois que l'on a obtenu un codage canonique de taille linéaire pour chacun des deux tournois, il suffit de comparer les deux codages. Cela peut être fait en temps $O(n)$.

Deuxième partie

Algorithmes Efficaces.

CHAPITRE 4

Composantes de Chevauchements

Plan

1. Introduction et Notations

Dans ce chapitre nous présentons une technique efficace pour calculer les composantes de chevauchements. Le problème considéré est le suivant :
Soit $\mathcal{X}$ un ensemble fini, et soit $\mathscr{F} = \{X_1, \ldots, X_p\}$ une famille de sous-ensembles de $\mathcal{X}$. Nous ne faisons pas d'hypothèse particulière sur la nature de $\mathscr{F}$. Le problème consiste à calculer pour $\mathscr{F}$ quelles sont les composantes de chevauchements pour la famille.

Le calcul des composantes de chevauchement est un problème important en algorithmique il est notamment utilisé pour la reconnaissance des graphes de parités Burlet et Uhry (1982). Cette procédure intervient également dans le problème du calcul efficace de la décomposition en coupes (*cf.* définition 1.6 *p.* 5) ou encore dans la reconnaissance des matrices ayant la propriété des 1 consécutifs McConnell (2004), pour ce problème cette procédure est utile pour exhiber un certificat.

Cette procédure est même utilisé dans cette thèse pour trouver les atomes d'une famille de parties au Chapitre. 2.

L'algorithme que nous présentons est dérivé de l'algorithme présenté par Dahlhaus (2000).

Définition 4.1 (Graphe de Chevauchement). Soit $OG(\mathscr{F}, E)$ le graphe de chevauchement de la famille $\mathscr{F}$. Les sommets représentent les éléments de la famille. Il y a une arête entre u et $v$si et seulement si $u \owedge v$.

DÉFINITION 4.2 (Composantes de Chevauchements). Les composantes de chevauchements d'une famille $\mathscr{F}$ forment une partition de $\mathscr{F}$ où chacune des parties correspond à une composante connexe du graphe de chevauchement OG.

Le problème considéré consiste donc à trouver cette partition de $\mathscr{F}$. L'algorithme que nous allons présenter s'exécute en temps linéaire. Cependant vu la nature de la donnée il convient de préciser ce que nous entendons par linéaire. La donnée est la famille $\mathscr{F}$ définie sur l'ensemble X. Par convention nous noterons $n = |\mathcal{X}|$ et $|\mathscr{F}| = \sum_{i=1}^{p} X_i$. Afin de ne pas alourdir plus encore les notations, nous utiliserons f pour désigner $|\mathscr{F}|$.

L'algorithme que nous présentons ici admet une complexité en temps $O(n + f)$.

2. Algorithme de Dahlhaus

Une première approche pour ce problème, serait de calculer le graphe de chevauchement OG et ensuite de calculer simplement ses composantes connexes. Cependant, nous rencontrons un problème de taille ! En effet, considérons la famille $\mathscr{F} = \{\{x_1, x_2\}, \{x_1, x_3\}, \ldots, \{x_1, x_p\}\}$. Le graphe OG comporte exactement p sommets et $|E(G)| = p(p-1)/2 = \Theta(p^2)$ ce qui est quadratique en f (*i.e.* $O(n + f^2)$).

L'approche de Dahlhaus est très surprenante dans la mesure où au lieu de calculer un sous-graphe OG' de OG qui aurait une taille linéaire en n et f, il construit un graphe complètement différent $D = (\mathscr{F}, L)$, où bien sûr l'ensemble des sommets est le même mais l'ensemble des arêtes peut différer complètement du graphe de chevauchement. Néanmoins, une des propriétés fortes de D est que ses composantes connexes correspondent exactement aux classes de chevauchements.

Soit LF la liste de tous les éléments $X_i \in \mathscr{F}$ triés par ordre décroissant selon leurs tailles. L'ordre des ensembles de taille identique est fixé arbitrairement. Étant donné un élément X de $\mathscr{F}$, nous noterons par $Max(X)$ l'élément Y plus grand que X (*i.e.* $|Y| \geqslant |X|$) tel que $Y \circledcirc X$. Remarquons que $Max(X)$ peut ne pas être défini pour certains éléments de $\mathscr{F}$. Dans ce dernier cas, et afin de simplifier la présentation de certains points techniques nous noterons $Max(X) = \varnothing$.

L'algorithme de Dahlhaus (2000) est basé sur l'observation fondamentale qui suit :

LEMME 4.3 (Dahlhaus (2000)). *Soit X un élément de $\mathscr{F}$ tel que $Max(x) \neq \varnothing$. Alors pour tout $Y \in \mathscr{F}$ tel que $Y \cap X \neq \varnothing$ et $|X| \leqslant |Y| \leqslant |Max(X)|$, $Y \circledcirc X$ ou $Y \circledcirc Max(X)$.*

DÉMONSTRATION. Si Y ne chevauche pas X, comme nous avons $|X| \leqslant |Y|$ et $Y \cap X \neq \varnothing$ cela signifie que $X \subseteq Y$. En conséquence de quoi $Y \cap Max(X) \neq \varnothing$. Par conséquent si Y ne chevauche pas $Max(X)$, alors $Max(X) \subseteq Y$. Mais dans ce cas comme nous avons $|Y| \leqslant |Max(X)|$, donc $Y = Max(X)$ et chevauche X. Donc finalement Y chevauche X ou $Max(X)$. □

Considérons que nous avons déjà calculé tous les $Max(X)$, par la suite nous verrons en section 3 plusieurs méthodes de calcul. Pour chaque élément x de $\mathcal{X}$ nous calculons la liste $SL(x)$ de tous les éléments X de $\mathscr{F}$ qui contiennent x. Cette liste est trié par ordre croissant sur la taille des éléments.

Calculer et trier toutes les listes pour tous les éléments x de $\mathcal{X}$ peut être fait globalement en temps linéaire, $O(n+f)$, en utilisant un tri par "bucket" global.

Le Graphe de Dahlhaus $D(\mathscr{F}, L)$ est construit à partir de ces listes. Soit X un ensemble contenant x tel que $Max(X) \neq \varnothing$. Pour tous les éléments consécutifs Y W après X dans $SL(v)$ (X compris) et tel que $|Y| \leqslant |Max(X)|$, nous créons dans D un arête YW.

Nous devons maintenant nous convaincre que le graphe D admet bien les mêmes composantes connexes que OG, c'est le but du lemme suivant.

LEMME 4.4 (Dahlhaus (2000)). *Les graphes $D = (\mathscr{F}, L)$ et $OG = (\mathscr{F}, E)$ ont les mêmes composantes connexes.*

DÉMONSTRATION. ($\Rightarrow$) Soient Y et W deux éléments de $\mathscr{F}$ tel que $YW \in L$. Par construction il existe un élément x de $\mathcal{X}$ tel que Y et W sont consécutifs dans $SL(x)$ et il existe un élément X de $\mathscr{F}$ qui apparaît avant Y et W dans $SL(x)$ tel que $Max(X) \neq \varnothing$ et que $|X| \leqslant |Y| \leqslant |W| \leqslant |Max(X)|$. Le lemme 4.3 nous dit que Y et W chevauchent soit X soit $Max(X)$. Comme X et $Max(X)$ se chevauchent, les éléments X, Y, W et $Max(X)$ appartiennent à la même composante de chevauchement que celle de $OF(\mathscr{F}, E)$.

Par extension, les sommets d'un chemin connexe dans $D = (\mathscr{F}, L)$ appartiennent à la même classe de chevauchement que celle de $OG = (\mathscr{F}, E)$.

($\Leftarrow$) Soient A et B deux éléments de $\mathscr{F}$ qui se chevauchent *i.e.* $AB \in E$. Et soit $v \in A \cap B$, supposons sans perte de généralité que $|A| \leqslant |B|$. Alors $Max(A) \neq \varnothing$ et $|Max(A)| \geqslant |B|$. De plus dans $SL(x)$, il existe une série de paires consécutives de A à B qui sont connectées dans $D = (\mathscr{F}, L)$. Par conséquent, A et B sont connectés dans $D = (\mathscr{F}, L)$. □

Remarquons que l'ordre des ensembles de tailles égales n'a aucune importance pour la construction du graphe de Dahlhaus. La figure 4.1 présente un exemple d'un graphe de chevauchement et du graphe de Dahlhaus.

LEMME 4.5 (Dahlhaus (2000)). *Ayant tous les $Max(X)$ pour tout X de $\mathscr{F}$, nous pouvons construire le graphe $D = (\mathscr{F}, L)$ en temps $O(n+f)$, et son nombre d'arêtes est de l'ordre de $O(f)$.*

DÉMONSTRATION. Afin de construire le graphe $D = (\mathscr{F}, L)$ à partir des listes $SL(x)$, il suffit de parcourir chacune d'elles du plus petit ensemble (en terme de cardinalité) au plus grand et se souvenir à chaque étape,le plus grand $Max(X)$ déjà rencontré. Si la taille

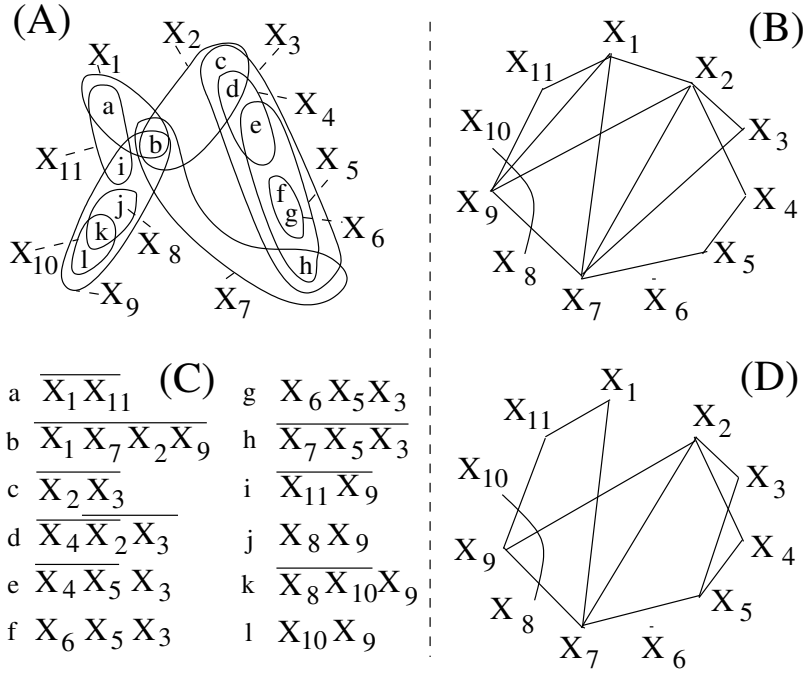


Figure 4.1. Exemple global(A) Une famille de sous-ensembles composée de 11 éléments, (B) Le graphe de chevauchement, (C) Les listes SL, (D) Le graphe de Dahlhaus. Sur (C) les intervalles définis par $Max(X)$ sont surlignés.

de l'élément courant est plus petite ou égale à cette valeur, une arête est créée entre les deux derniers ensembles considérés.

Examinons maintenant le nombre d'arêtes de $D = (\mathscr{F}, L)$. Comme au plus une arête est créée pour chaque ensemble dans un liste $SL(x)$, au plus f arêtes sont créées après le traitement de toutes les listes SL. □

Une fois que ce graphe est construit, l'identification des composantes de chevauchements se fait à l'aide d'un parcours de graphe classique (au choix en profondeur DFS, ou en largeur BFS) sur le graphe D. La complexité obtenue est donc $O(n + f)$ en temps et en espace.

Il reste maintenant à montrer que nous pouvons calculer tous les $Max(X)$ de manière efficace.

3. Calcul des Max(X)

Soit LF La liste de tous les éléments X de $\mathscr{F}$ triée par ordre décroissant selon la taille. Nous considérons également la matrice booléenne BM de taille $f \times n$ définie de la manière suivante :

Chaque ligne représente un élément X de $\mathscr{F}$ selon l'ordre de LF, et chaque colonne représente un élément x de $\mathcal{X}$. La valeur de $BM[i,j]$ vaut 1 si $x_j \in X_i$.

La première étape de l'algorithme de Dahlhaus consiste à trier les colonnes de BM selon l'ordre lexicographique. Dans le papier original, l'auteur ne précise pas comment procéder pour effectuer ce tri. Nous détaillerons la procédure que nous avons utilisé plus tard. Nous considérons donc que les colonnes de cette matrice sont triées selon l'ordre lexicographique. La figure 4.2 présente la matrice booléenne associée à la famille présentée en figure 4.1.

	1 a	2 i	3 l	4 j	5 k	6 b	7 c	8 d	9 h	10 f	11 g	12 e	left	right
X_3	0	0	0	0	0	0	1	1	1	1	1	1	7	12
X_9	|	1	1	1	1	1	0	0	0	0	0	0	2	6
X_5	|	0	0	0	0	0	0	0	1	1	1	1	9	12
X_2	0	|	|	|	|	1	1	1	0	0	0	0	6	8
X_1	1	|	|	|	|	1	0	0	|	|	|	0	1	6
X_4	0	|	|	|	|	0	|	1	|	0	0	1	8	12
X_6	|	|	|	|	|	0	|	0	0	1	1	0	10	11
X_7	|	|	|	0	0	1	|	|	1	0	0	|	6	9
X_8	|	|	0	1	1	0	|	|	0	|	|	|	4	5
X_{10}	0	0	1	0	1	0	|	|	|	|	|	|	3	5
X_{11}	1	1	0	0	0	0	0	0	0	0	0	0	1	2

Figure 4.2. La suite de la figure 4.1 : La matrice BM associée à la famille. Les lignes sont triées par ordre décroissant selon la taille de $X \in \mathscr{F}$ et les colonnes sont triées par ordre lexicographique.

Pour chaque élément X de $\mathscr{F}$ nous noterons par $left(X)$ (*resp.* $right(X)$) le numéro de la colonne qui contient le 1 le plus à gauche dans BM (*resp.* le plus à droite) dans la ligne de X.

LEMME 4.6. *Soient X et Y des éléments de $\mathscr{F}$ tels que $X \circledcirc Y$ et soit r_Y le numéro de la ligne correspondante à Y dans BM. Alors il existe un ligne t plus haute ou égale à r_Y telle que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$*

DÉMONSTRATION. Comme Y chevauche X, $|X| \geqslant 2$. Soit r_X la ligne correspondante à X dans BM. Maintenant comme $X \circledcirc Y$, existent deux indices $1 \leqslant i \leqslant j \leqslant n$ et une ligne r tels que $BM[r_X, i] = BM[r_X, j] = 1$ telle que une des valeurs de $BM[r, i]$ et $BM[r, j]$ est égale à 1 et l'autre à 0.

Nous considérons maintenant le plus grand r qui satisfait ces conditions.

Dans un premier temps, si $BM[r,i] = 1$ et $BM[r,j] = 0$, alors comme $i < j$ et comme toutes les colonnes ont été triées par ordre lexicographique, il doit exister une ligne r' plus haute que r telle que $BM[r',i] = 0$ et $BM[r',j] = 1$. Maintenant, nous considérons sans perte de généralité, que $BM[r,i] = 0$ et $BM[r,j] = 1$

Parmi tous les couples d'indices i et j tels que $BM[r_X,i] = BM[r_X,j] = 1$ et il existe r telle que $BM[r,i] = 0$ et $BM[r,j] = 1$, considérons un couple i' et j' avec $1 \leqslant i' \leqslant j' \leqslant n$, qui est associé à la plus haute ligne que nous noterons t.

Montrons maintenant que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$. Si $BM[t, left(X)] = 1$, par conséquent $i > left(X)$ et comme $BM[t,i] = 0$ et que les colonnes sont triées par ordre lexicographique, il devrait exister un ligne r' plus haute telle que $BM[r', left(X)] = 0$ et $BM[r',i] = 1$, ce qui contredit le fait que t soit la plus haute ligne avec ces propriétés. Par conséquent $BM[t, left(X)] = 0$. De manière symétrique, le même argument est valable pour montrer que $BM[t, right(X)] = 1$. □

LEMME 4.7. *Soit X un élément de $\mathscr{F}$. Alors $Max(X) \neq \varnothing$ si et seulement si il existe une ligne t de BM telle que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$ correspondant à l'élément Y de $\mathscr{F}$ vérifiant $|Y| \geqslant |X|$.*

DÉMONSTRATION. ($\Leftarrow$) Si un ensemble Y correspond à une ligne t dans BM tel que $BM[t, left(X)] = 0$. et $BM[t, right(X)] = 1$, Y chevauche trivialement X. Comme $|Y| \geqslant |X|$, $Max(X) \neq \varnothing$.
($\Rightarrow$) Supposons que $Max(X) \neq \varnothing$ et soit r_M sa ligne correspondante dans BM. Donc par le lemme 4.6, il existe une ligne t dans BM telle que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$ et telle que t est plus haute ou égale à r_M. Comme $Max(X)$ vérifie que $|Max(X)| \geqslant |X|$, l'ensemble correspondant à r_M est aussi tel que $|Y| \geqslant |X|$. □

LEMME 4.8 (Dahlhaus (2000)). *Soit X un élément de $\mathscr{F}$ tel que $Max(X) \neq \varnothing$. Alors $Max(X)$ correspond à la plus haute ligne t dans BM telle que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$.*

Remarquons la ligne $Max(X)$ peut être plus basse dans la matrice BM que la ligne de X. Sur l'exemple (*cf.* FIG. 4.1 et 4.2) , ce cas se présente pour X_8 et X_{10}, en effet $Max(X_{10}) = X_8$ mais aussi $Max(X_8) = X_{10}$.

DÉMONSTRATION. Supposons que $Max(X) \neq \varnothing$ et soit r_M cette ligne dans BM. Alors en appliquant le lemme 4.6, il existe un ligne t dans BM telle que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$ et telle que t est plus haute ou égale à r_M. Cependant comme une telle ligne t correspond à un ensemble qui chevauche X et que $Max(X)$ est le plus grand (en terme de cardinalité) de ces ensembles dans la liste LF, nous en concluons que $t = r_M$. □

Si on applique ce lemme à notre exemple (FIG. 4.2), nous obtenons que $Max(X_1) = X_9$ car $left(X_1) = 1$ et $right(X_1) = 6$ et X_9 (à la ligne 2) est la ligne la plus haute avec 0 en première colonne et 1 en sixième colonne.

L'approche utilisée par Dahlhaus pour calculer tous les $Max(X)$ est d'identifier pour chaque ligne r correspondante à X la plus haute ligne t (*cf.* 4.8) telle que $BM[t, left(X)] = 0$ et $BM[t, right(X)] = 1$. Afin de calculer cela efficacement, Dahlhaus réduit le problème à calculer un LCA (Plus petit Ancêtre Commun). Nous allons expliquer sa solution en section 3.1. Nous présenterons par la suite, section 3.2 une technique que nous avons développé qui utilise des affinages de partition.

Cette nouvelle approche est bien plus simple à implémenter que le calcul du LCA en temps linéaire. De plus, notre approche permet de calculer assez simplement l'ordre lexicographique sur les colonnes.

3.1. Calcul des Max(X) avec le LCA. Nous considérons maintenant toutes les colonnes intermédiaires entre toute paire de colonnes dans BM. Dans ces colonnes, pour chacune des lignes, nous plaçons un point • entre les motifs *"01"* et *"10"*. Une illustration est faite en figure 4.3(a).

Nous relions les points les plus élevées dans l'arbre de Dahlhaus de la manière suivante :

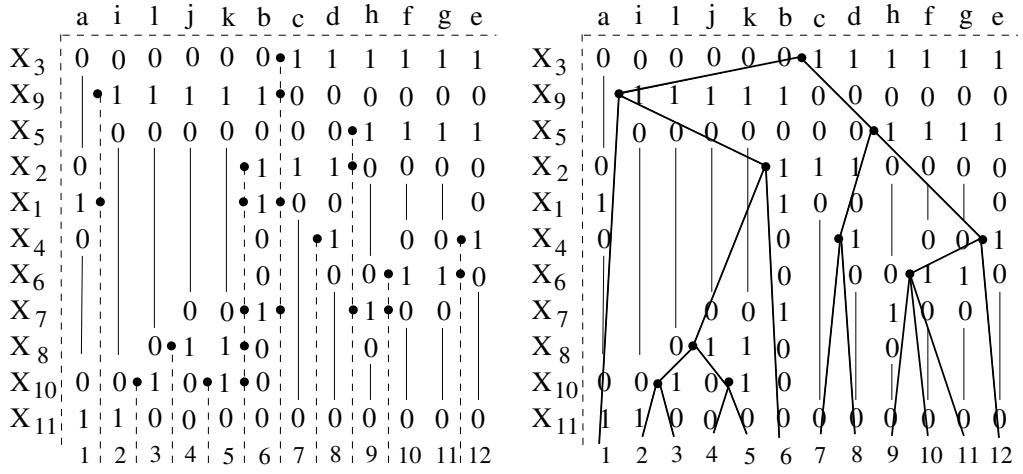
(1) La racine de l'arbre est le point le plus haut . De plus il ne peut y avoir qu'un seul point le plus élevé. De plus il doit y avoir une racine s'il existe un élément X de $\mathscr{F}$ si X diffère par au moins élément de $\mathcal{X}$.

(2) Nous effectuons récursivement le même processus : Chaque nouveau point *np* dans l'arbre (racine incluse) coupe la sous-matrice en deux, selon les colonnes intermédiaires ou il est placé. L'enfant gauche (*resp.* droit) de *np* est le plus haut point dans la partie gauche de la matrice (*resp.* droite), si toutefois ce point existe. Notons que l'ordre lexicographique des colonnes de BM nous assure qu'il peut y avoir au plus un seul point le plus haut dans chacune des sous-parties.

(3) Quand une sous-partie ne contient aucun nouveau point, une feuille par colonne de cette sous partie est créée et est attachée comme enfant au point qui a crée cette sous-partie. Si la feuille est à gauche du point (parent), alors la feuille est marqué comme enfant gauche, sinon comme enfant droit. Chaque feuille est étiquetée avec le numéro de la colonne dans BM.

Une illustration de cette construction est donné dans la figure 4.3(b).

PROPOSITION 4.9 (Dahlhaus (2000)). *Soit X un élément de $\mathscr{F}$, et soit Y élément de $\mathscr{F}$ qui correspond à la ligne de $LCA(left(X), right(X))$ dans BM. Si $|Y| \geqslant |X|$, alors $Y = Max(X)$. Autrement $Max(X) = \varnothing$.*



(a) Placement des points. (b) L'arbre de Dahlhaus.

Figure 4.3. Suite de FIG. 4.1 et 4.2 : Arbre construit à partir de la matrice BM.

DÉMONSTRATION. Soit r le numéro de la ligne de $LCA(left(X), right(X))$ dans BM et soit l la position de la colonne dans BM qui est juste avant le point représentant $LCA(left(X), right(X))$.

Tout d'abord, $BM[r, l] = 0$ et $BM[r, l+1] = 1$. Supposons qu'au contraire que $BM[r, l] = 1$ et $BM[r, l+1] = 0$. Comme toutes les colonnes de BM sont triées par ordre lexicographique, il doit exister une ligne r' telle que $BM[r', l] = 0$ et $BM[r', l+1] = 1$ et par conséquent un point doit exister entre les colonnes l et $l+1$ qui est plus haute que r, ce qui est en contradiction avec la construction de l'arbre DT.

Nous démontrons maintenant que $BM[r, left(X)] = 0$ et $BM[r, right(X)] = 1$. Supposons qu'au contraire $BM[r, left(X)] = 1$. Alors, une fois de plus, comme les colonnes sont triées par ordre lexicographique, il doit exister une ligne r' plus haute que r telle que $BM[r', left(X)] = 0$ et $BM[r', l] = 1$. Ce qui, une fois de plus, constitue une contradiction avec la construction de l'arbre DT. Cette technique marche également pour le coté droit.

Nous prouvons ensuite que r est la ligne la plus haute à satisfaire cette propriété. Supposons le contraire, à savoir qu'il existe r' telle que $BM[r', left(X)] = 0$ et $BM[r', right(X)] = 1$. Alors, il devrait y avoir une coupure 01 quelque part dans cette ligne qui aurait séparé $left(X)$ et $right(X)$. Ce qui implique qu'il aurait dû y avoir un noeud dans DT dans une ligne supérieure ou égale à r' qui aurait coupé $left(X)$ et $right(X)$, ce qui contredit que r est la ligne qui correspond à $LCA(left(X), right(X))$.

Si $|Y| \geqslant |X|$, par le lemme 4.8 $Max(X) \neq \varnothing$ et l'ensemble Y qui correspond à r est tel que $Y = Max(X)$.

Si $|Y| < |X|$, comme aucune ligne r' plus haute que r peut vérifier $BM[r', left(X)] = 0$ et $BM[r', right(X)] = 1$ par le lemme 4.7 $Max(X) \neq \varnothing$. □

Par exemple, sur l'exemple commencé en figure 4.1, X_9 correspond à la ligne $LCA(1,2) = LCA(left(X_{11}), right(X_{11}))$. Et comme $|X_9| \geqslant |X_{11}|$, $X_9 = Max(X_{11})$.

3.2. Calcul des Max(X) par affinage de partition. Nous présentons maintenant une technique alternative pour calculer les $Max(X)$ qui ne se sert pas du LCA. Cette technique est fait appel à un paradigme désormais bien connu en algorithmique, qui est l'affinage de partition, on pourra consulter les travaux de Paige et Tarjan (1987) sur le sujet.

Les données que nous manipulons sont des partitions ordonnées de $\mathcal{X}$ que nous affinons avec chaque élément X de $\mathscr{F}$ qui sont pris dans l'ordre de LF (*i.e.* par taille décroissante). La partition initiale est l'ensemble $\mathcal{X}$ en entier, que nous notons avec la notation usuelle $\mathcal{P}_{\mathcal{X}}$.

Nous rappelons qu'un ensemble d'une partition est appelé partie de la partition. Et dans chaque partition l'ordre des parties est important, mais l'ordre des éléments d'une même partie n'a aucune importance. Soit $C = \{x_1, \ldots, x_k\}$ une partie dans une partition. Affiner C avec $X \in \mathscr{F}$ consiste à extraire tous les éléments v_i de C qui appartiennent à X et de créer une nouvelle partie C'', et C' le reste $C \backslash C''$. En d'autres termes, $C' = (C \setminus X) = (C \backslash C'')$ et $C'' = C \cap X$. Et comme, rappelons le, l'ordre des parties est important, nous remplaçons, C par $C'C''$. Dans le cas où C ne contient que des éléments de X ou aucun, évidemment C reste inchangé.

Affiner une partition $\mathcal{P}$ avec un ensemble X consister à affiner toutes les parties de $\mathcal{P}$ avec X. Nous notons cela par $\mathcal{P}_{|X}$.

Notre approche procède en trois étapes :

(1) Affiner $\mathcal{P}_{\mathcal{X}}$ avec tous les éléments X de $\mathscr{F}$ pris dans l'ordre de LF.

(2) Calculer ensuite pour chaque X de $\mathscr{F}$ les valeurs de $left(X)$ et $right(X)$. Trier ensuite tous les éléments X de $\mathscr{F}$ vis à vis de ces valeurs.

(3) Affiner une fois encore $\mathcal{P}_{\mathcal{X}}$ pris dans l'ordre de LF mais en tenant compte des informations calculées à l'étape (2) pour calculer tous les $Max(X)$.

Nous détaillons maintenant chacune des étapes :

Étape 1 - Affiner $\mathcal{P}_{\mathcal{X}}$. Z Considérons la partition finale que nous obtenons après avoir affiné $\mathcal{P}_{\mathcal{X}}$ avec tous les éléments X de $\mathscr{F}$ pris dans l'ordre de LF. Nous notons cette partitions $\mathcal{P}_F$.

LEMME 4.10. *La partition $\mathcal{P}_F$ donne un ordre lexicographique des colonnes de BM.*

DÉMONSTRATION. L'affinage de partition consiste à trier lexicographiquement une ligne de BM en ne touchant seulement que les 1 mais également en conservant l'ordre définie par les affinages précédents. Par conséquent, affiner $\mathcal{P}_\mathcal{X}$ selon l'ordre de LF consiste à trier lexicographiquement les colonnes de BM de haut en bas. □

Dans notre exemple, sur les données de la figure 4.1 $\mathcal{P}_F = \{a\}\{i\}\{l\}\{j\}\{k\}\{b\}\{c\}\{d\}\{h\}\{f,g\}\{e\}$. Remarquons également que les colonnes qui sont égales dans BM sont dans la même partie de $\mathcal{P}_F$, pour lesquelles nous fixons un ordre arbitraire.

Étape 2 - Calculer les valeurs left(X) et right(X). Ensuite nous calculons les valeurs $left(X)$ et $right(X)$ dans la partition finale $\mathcal{P}_F$. La complexité pour effectuer cette opération est en temps $O(n+f)$. Pour cela il suffit de parcourir chacun des éléments X de $\mathscr{F}$ et pour chacun se souvenir de l'indice maximum et minimum selon l'ordre donné par la partition $\mathcal{P}_F$. Nous mettons aussi à jour une structure de donnée AM qui pour chaque position i contient la liste des éléments X de $\mathscr{F}$ pour lesquels $right(X) = i$. Toutes ces listes sont triées par ordre croissant selon la valeur de $left(X)$.

Cette structure de données nous permet également de supprimer un élément X de $\mathscr{F}$ de la liste $AM[right(X)]$ en temps $O(1)$. Pour ce faire nous pouvons utiliser une liste doublement chainée pour coder chacune des listes. Cette structure peut facilement être construite de manière globale en procédant à un tri par *"bucket"*. La complexité globale est en temps $O(n+p)$, où, rappelons-le, p est le nombre d'éléments de la famille $\mathscr{F}$.

Étape 3 - Affiner $\mathcal{P}_\mathcal{X}$ une fois de plus et calcul des Max(X). L'idée principale est la suivante. Supposons qu'à un étape du processus d'affinage de partitions selon l'ordre de LF, nous affinons une partie $C = \{x_1, \ldots, x_k\}$ de la partition $\mathcal{P}$ par Y un élément de $\mathscr{F}$. Et le résultat obtenu après cette affinage est $C \to C'C''$.

LEMME 4.11. *Soit X un élément de $\mathscr{F}$ tel que $|X| \leqslant |Y|$, avec $left(X) \in C'$ et $right(X) \in C''$. Alors, $Y = Max(X)$.*

Remarquons que si $|X| = |Y|$, alors X pourrait être avant Y dans LF.

DÉMONSTRATION. Soit r la ligne correspondante à Y dans BM. Comme $left(X) \in C'$ et $right(X) \in C''$, alors $BM[r, left(X)] = 0$ et $BM[r, right(X)] = 1$, et Y chevauche X trivialement. De plus nous avons $|X| \leqslant |Y|$, donc $Max(X) \neq \varnothing$. Avec cela, la ligne r est la plus haute ligne telle que $BM[r, left(X)] = 0$ et $BM[r, right(X)] = 1$, car autrement les éléments de X auraient été coupés par un ensemble plus grand que Y dans l'ordre LF. Donc par le lemme 4.8, nous obtenons $Y = Max(X)$. □

La dernière phase de l'algorithme consiste donc à affiner $\mathcal{P}_\mathcal{X}$ une fois de plus avec les tous les éléments Y pris selon l'ordre de la liste LF.

Dans un premier temps, nous initialisons les valeurs de $Max(X)$ à $\varnothing$. Chaque fois que de nouvelles parties $C'C''$ apparaissent (à savoir entre les positions l et $l+1$), pour tout éléments x de C'', toutes les listes $AM[v]$ sont inspectées de la manière suivante :
Soit X le début (top) d'une de ces listes, Tant que $left(X) \leqslant l$, X est supprimé de la liste et $Max(X) \leftarrow Y$. Après avoir affiné avec Y, s'il n'y a plus de Y' tels que $Y' \leqslant_{LF} Y$ et $|Y'| = |Y|$, tous les ensembles de même taille que Y sont supprimés de la structure AM.

LEMME 4.12. *Notre algorithme calcule correctement à l'étape 3 tous les $Max(X)$, pour tout X de $\mathscr{F}$.*

DÉMONSTRATION. À l'étape 1, l'ordre lexicographique des colonnes de BM est calculé. Le résultat est donné par la partition "finale" $\mathcal{P}_F$ (*cf.* lemme 4.10).

Une fois l'étape 2 effectuée, toutes les valeurs $left(X)$ et $right(X)$, pour tout X de $\mathscr{F}$, sont calculées et la structure de données AM est créée.

À l'étape 3, la correction du calcul repose sur l'observation suivante : pour chaque nouvelle partition $\mathcal{P}$ créée après un affinage, tous les ensembles X qui demeurent dans AM sont tels que $left(X)$ et $right(X)$ appartiennent à la même partie de $\mathcal{P}$. Cela est trivialement vérifié car sinon les ensembles auraient été coupés par un affinage précédent et supprimé de la structure AM. Cela a pour conséquence qu'après une coupe d'une partie C en $C'C''$ par un ensemble Y, tester si $left(X) \in C'$ et $right(X) \in C''$ pour tous les éléments présents dans AM est équivalent à tester si $right(X) \in C''$ et si $left(X) \leqslant l$, où l est la position gauche de la coupe dans $\mathcal{P}$ entre C' et C''. De plus, comme chaque ensemble est traité selon l'ordre de LF et est utilisé pour un possible affinage. Il est supprimé de AM une fois que les ensembles de taille identiques ont été traités. Quand un ensemble Y coupe une partie C en $C'C''$, tous les ensembles dans AM respectent $|X| \leqslant |Y|$.

Par conséquent, toutes les conditions du lemme 4.11 sont satisfaites. Donc si une valeur $Max(X)$ est affectée par notre algorithme, elle est affectée avec la valeur de droite.

Supposons maintenant qu'un ensemble X admette un ensemble Y comme $Max(X)$. Il est assuré qu'à une certaine étape de l'algorithme Y a été désigné comme $Max(X)$ comme par définition $|X| \leqslant |Y|$ ce qui implique que X est encore dans AM quand Y est pré-traité et que par le lemme 4.8 $left(X) \notin Y$ et $right(X) \in Y$. L'ensemble Y a donc coupé une partie C en $C'C''$ tel que $right(X) > l$ et $left(X) \leqslant l$. □

Il reste maintenant à expliquer comment nous implémentons de manière efficace la procédure d'affinage de partitions. Nous utilisons le fait qu'à l'intérieur d'une partie, l'ordre des éléments n'importe pas pour obtenir une implémentation simple : une partition est représentée comme une tableau de taille n dans lequel chacune des cellules contient (a) un élément de $\mathcal{X}$ et (b) un pointeur vers la partie de la partition dans laquelle il est contenu.

Un partie est représentée comme une paire d'entiers qui représentent les bornes dans cette table, une illustration de cette structure est donnée en figure 4.4.

Affiner une partition $\mathcal{P}$ avec un ensemble Y peut être fait en temps $O(|Y|)$ de la manière suivante. Soient $[i, j]$ les bornes d'une partie C telle que $C \subsetneq Y$ (facile à tester). Soit k le nombre d'éléments de Y qui appartient à la sous-table $[i, j]$, $1 \leqslant k \leqslant j - i$. Nous échangeons les éléments de la sous-table $[i, j]$ pour placer les k éléments de Y à la fin de la sous-table. Nous ajustons ensuite les bornes de C à $[i, j - k]$ et nous créons une nouvelle partie $[j - k + 1, j]$ vers laquelle les k éléments de Y pointent désormais.

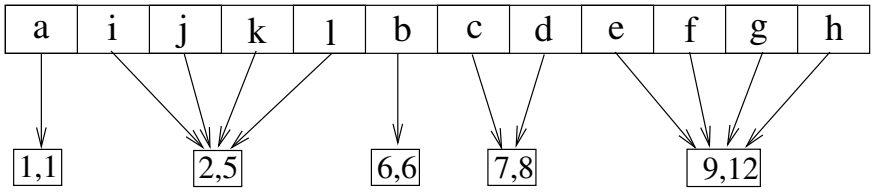


Figure 4.4. Suite de l'exemple : Implémentation de la partition $\mathcal{P} = \{a\}\{i, j, k, l\}\{b\}\{c, d\}\{e, f, g, h\}$

THÉORÈME 4.13. *L'identification des composantes de tous les $Max(X)$, pour tout X de $\mathscr{F}$, en utilisant l'affinage de partition peut être fait en temps $\Theta(n + f)$.*

DÉMONSTRATION. Par le lemme 4.12 l'algorithme est correct. Les étapes 1 et 2 s'effectuent en un temps $O(n + f)$. Lors de l'étape 3, le fait que toutes les listes dans AM soient triées par ordre croissant sur les valeurs de $left(X)$ garanti que quand un ensemble Y coupe une partie C en $C'C''$, identifiant et supprimant tous les ensembles X tels que $left(X) \in C'$ et $right(X) \in C''$ peut être fait en temps $\Theta(|C| + |K| + 1)$, où K est le nombre de ces ensembles. Supprimer un ensemble de AM se fait en temps constant, par conséquent le temps total pour gérer AM est en temps $\Theta(n + f)$. □

Une implémentation de cet algorithme a été écrite en langage C par M. Rao et est disponible à l'adresse suivante `http://www.liafa.jussieu.fr/~raffinot/overlap.html`.

4. Calcul d'un Sous-graphe du graphe de chevauchement

Certaines applications comme celle présentée par McConnell (2004) ont besoin d'obtenir un sous-arbre couvrant du graphe de chevauchement OG. Or, dans notre cas comme dans celui de Dahlhaus (2000), nous obtenons un graphe qui "code" les composantes de chevauchement, mais le graphe obtenu n'est pas pour autant un sous graphe du graphe de chevauchement.

L'approche développée par McConnell (2004), pour trouver un tel arbre, consiste à calculer en premier lieu le graphe de Dahlhaus DG et procède ensuite à des modifications locales sur ce graphe. Son approche est pour le moins compliquée.

Nous présentons ici, une manière de modifier l'algorithme de Dahlhaus afin qu'il renvoie un sous-graphe du graphe OG. La taille du sous-graphe ainsi calculé reste linéaire. Les composantes connexes de ce graphe correspondent toujours aux composantes de chevauchements. Ainsi, une fois ce graphe obtenue, il est aisé d'obtenir un sous-arbre (forêt) couvrant du graphe OG. L'idée de modification est la suivante :

LEMME 4.14. *Soient X et Y des éléments de $\mathscr{F}$ tels que $X \cap Y \neq \varnothing$, tels que $Max(X) \neq \varnothing$ et tels que $|X| \leqslant |Y| \leqslant |Max(X)|$. Soit r_Y la ligne de Y dans BM. Si $BM[r_Y, left(X)] = 0$, Y chevauche X. Sinon, (a) si $BM[r_Y, right(X)] = 0$, alors Y chevauche X et (b) si $BM[r_Y, right(X)] = 1$ Y chevauche $Max(X)$.*

DÉMONSTRATION. Soit r_X la ligne de BM correspondante à l'ensemble X, et soit r la ligne correspondante à $Max(X)$. Si $BM[r_Y, left(X)] = 0$, comme $BM[r_Y, left(X)] = 1$ et que $X \cap Y \neq \varnothing$ et $|X| \leqslant |Y|$ nous en concluons que X chevauche Y.

Considérons maintenant que $BM[r_Y, left(X)] = 1$,
Dans le cas (a) : si $BM[r_Y, right(X)] = 0$, alors comme $BM[r_X, right(X)] = 1$, avec les mêmes arguments utilisés ci-dessus X chevauche Y. En ce qui concerne le cas (b) : si $BM[r_Y, right(X)] = 1$, alors avec le lemme 4.8, $BM[r_X, right(X)] = 1$ et $BM[r_X, left(X)] = 0$ et que $|Y| \leqslant |Max(X)|$, Y chevauche X. □

Nous modifions la construction du graphe de Dahlhaus de la manière suivante : Nous considérons toujours les intervalles $X \ldots YW \ldots$ sur la liste $SL(v)$ tel que $Max(X) \neq \varnothing$ et $|W| \leqslant |Max(X)|$. Mais au lieu de créer une chaîne $X - \ldots Y - W - \ldots$ dans $D(\mathscr{F}, L)$, nous créons une arête entre X et $Max(X)$ si elle n'est pas déjà présente. et une liste de quintuplet $(left(X), right(X), X, Y, Max(X))$, $(left(X), right(X), X, W, Max(X))$,... Pour tous les éléments qui constitue le chemin entre X et $Max(X)$.
Tous les quintuplets sont placés dans une liste LQ_1. Remarquons que si un élément appartient à deux intervalles, un unique quintuplet est formé avec l'intervalle le plus à droite.

Pour appliquer le lemme 4.14, il suffit pour chaque quintuplet $(left(X), right(X), X, Y, Max(X))$ de tester si Y appartient à $SL(\mathcal{P}_{F_{left(X)}})$. Si Y n'appartient pas, nous créons une arête XY. Sinon, nous testons si Y appartient à $SL(\mathcal{P}_{F_{right(X)}})$, si Y n'appartient pas nous créons une arête XY, sinon, c'est que Y appartient à $SL(\mathcal{P}_{F_{right(X)}})$ nous créons une arête $YMax(X)$.

Pour des questions de temps de calcul (complexité) nous avons besoin d'effectuer ces tests rapidement pour tous les quintuplets contenus dans LQ_1. Nous procédons en deux

étapes. Durant la première étape nous cherchons tous les Y dans $SL(\mathcal{P}_{F_{left(X)}})$. Si Y n'appartient pas, nous ajoutons le quintuplet à une seconde liste LQ_2. Pendant la seconde étape si LQ_2 n'est pas vide, pour tous les quintuplets de la forme $(left(X), right(X), X, Y, Max(X))$ dans LQ_2 nous cherchons Y dans $SL(\mathcal{P}_{F_{right(X)}})$.

Nous considérons par la suite que toutes listes $SL(v)$ sont triées selon l'ordre de LF au lieu d'être simplement trié par tailles croissantes. Afin de comparer efficacement la liste LQ_1 avec toutes les listes $SL(v)$, il suffit de trier la liste LQ_1 selon la valeur de $left(X)$ et de trier les quintuplets qui ont la même valeur $left(X)$ selon l'ordre dans LF de Y. Cela peut se faire en temps $O(n+f)$ en utilisant un tri par "bucket". La comparaison de LQ_1 et des tableaux $SL()$ peut également être fait en temps $O(n+f)$ en comparant simultanément n liste triées. La même approche est valable pour LQ_2.

Nous obtenons par conséquent :

THÉORÈME 4.15. *Un sous-graphe du graphe de chevauchement de $\mathscr{F}$ ayant les mêmes composantes peut être calculé en temps $O(n+f)$.*

DÉMONSTRATION. Le lemme 4.14 nous garantit que le nouveau graphe est un sous graphe de OG. Pour montrer qu'ils ont les mêmes composantes connexes, il suffit donc de prouver que si deux éléments de $\mathscr{F}$, A et B se chevauchent, il existe un chemin connectant A à B dans le sous-graphe. L'observation suivante est la base même de la preuve :

Soient $X \dots Y \dots Z$ triés par ordre croissant sur le même $SL(v)$, et tels que $|Y| \leqslant |Max(X)|$, $|Max(X)| \leqslant |Max(Y)|$ et $|Z| \leqslant |Max(Y)|$. Alors il existe un chemin entre tous les ensembles X, Y et Z dans le nouveau sous-graphe, car par construction X et $Max(X)$ sont reliés par une arête, Y est connecté à X ou à $Max(X)$, Y est connecté à $Max(Y)$ et finalement Z est relié à Y ou à $Max(Y)$.

Considérons maintenant un élément $x \in A \cap B$. Supposons sans perte de généralité que $|A| \leqslant |B|$. Alors $Max(A) \neq \varnothing$ et $|Max(A)| \geqslant |B|$. Par ailleurs, dans $SL(v)$, il existe un séquence (éventuellement vide) de k ensembles $A \dots Y_1 \dots Y_2 \dots Y_k \dots B$ telle que $|B| \leqslant |Max(Y_k)|$, $|Y_k| \leqslant |Max(Y_{k-1})|$, et $|Y_1| \leqslant |Max(A)|$. En procédant par induction sur la séquence en utilisant l'observation précédente, il existe un chemin de A à B dans le sous-graphe.

Le sous-graphe peut trivialement être construit en temps $O(n+f)$ car toutes les étapes admettent cette complexité. □

Un illustration est donnée en figure 4.5

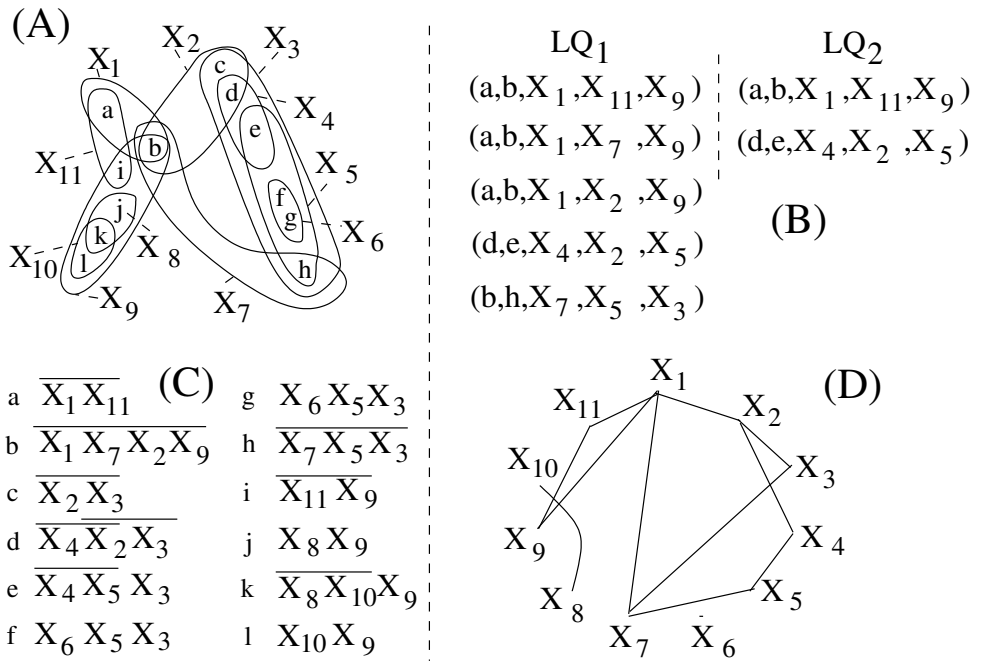


Figure 4.5. Exemple global

CHAPITRE 5

NLC-2

1. Introduction

Plan

La largeur NLC est un paramètre de graphe introduit par Wanke (1994). Cette notion est très étroitement reliée à la largeur de clique (clique-width) introduite par Courcelle et al. (1993). Ces deux paramètres furent introduit pour généraliser une décomposition de graphes bien connue : la largeur arborescente. Les recherches sur de telles largeurs ont pour motivation, que lorsque la largeur (NLC, de clique, arborescente, ...) est bornée par une constante, de nombreux problèmes **NP**-Complet peuvent être résolus en temps polynomial, voir même linéaire, pourvu que la décomposition soit donnée.

De tels paramètres donnent des informations structurelles sur le graphe concerné. Malheureusement, trouver une décomposition NLC minimum (*i.e.* la constante la plus petite possible) est un problème **NP**-Complet, ce résultat a été montré par Gurski et Wanke (2005).

Toutefois quelques résultats sont connus. Soit NLC-k la classe des graphes dont la largeur NLC est bornée par k. Les graphes de NLC-1 correspondent exactement aux cographes (*cf.* définition 1.25 *p.* 13). La classe des graphes NLC-2 constitue une généralisation des graphes NLC-1. Cette classe généralise un certain nombre de classes de graphes bien connues, telles que les probe-cographes, les bi-cographes ainsi que les weak bi-split et les

graphes totalement décomposables par bi-joins. Johansson a montré que reconnaître les graphes NLC-2 est un problème polynomial. Toutefois pour k supérieur à 2, le problème demeure ouvert.

Dans ce chapitre nous considérons les graphes de largeur NLC égale à 2. Nous fournissons un algorithme de reconnaissance de ces graphes en temps $O(n^2.m)$. Nous améliorons ainsi le résultat précédent de Johansson (2000) qui avait un algorithme en $O(n^4 log\ n)$. Par ailleurs, nous exhibons les liens étroits qui existent entre les graphes de largeur NLC 2 et trois décompositions de graphes présentées dans cette thèse, à savoir, la décomposition modulaire, la décomposition en coupes et la décomposition en bi-join.

La NLC 2 peut être définie comme un problème de coloration. Mais contrairement à la NLC k (pour $k \geqslant 3$) l'opération de recoloriage est inutile pour les graphes de largeur NLC égale à 2 et qui sont premiers pour la décomposition modulaire. À partir de cetter observation nous proposons une décomposition canonique des graphes de NLC 2 bi-colorés . Si la bi-coloration est fournie, nous pouvons clalculer cette décomposition canonique en temps $O(n.m)$. Si le graphe est premier pour la décomposition modulaire, en utilisant la décomposition en coupes et en bi-join sur le graphe, nous montrons qu'il y a au plus $O(n)$ bi-coloration à vérifier.

Finalement grâce à la décomposition modulaire nous montrons comment réduire le problème de reconnaître les graphes de largeur NLC égale à 2, au problème de reconnaître les graphes premier de largeur NLC 2. En section 3 nous présentons l'algorithme de décomposition en temps $O(n^2m)$.

En section 4, nous présentons un algorithme pour tester l'isomorphisme de deux graphes de largeur NLC 2 en temps polynomial. L'algorithme fait appel à la décomposition modulaire, la décomposition en coupes, la décomposition en bi-join et enfin à la décomposition canonique que nous avons introduit. Nous obtenons ainsi un algorithme en temps $O(n^2m)$.

2. Préliminaires

Soient A et B deux ensembles disjoints. Pour tout $(a, b) \in A \times B$ nous avons $\{a, b\} \in E$, et nous notons $A \circledcirc B$ si pour tout $(a, b) \in A \times B$ nous avons $\{a, b\} \notin E$.

Un *k-étiquetage* (ou *étiquetage*) est une fonction $l : V \to \{1, \ldots, k\}$.

Un graphe k-étiqueté est un graphe $G = (V, E, l)$ où V et E sont les ensembles usuels des sommets et des arêtes, et l est un k-étiquetage sur V. Nous le noterons indifféremment (G, l) ou (V, E, l).

Deux graphes étiquetés (V, E, l) et (V', E', l') sont isomorphes s'il existe une bijection $\varphi : V \to V'$ telle que $\{u, v\} \in E \Leftrightarrow \{\varphi(x), \varphi(y)\} \in E'$ et $l(u) = l'(\varphi(u))$ pour tout $u, v \in V$.

Soit k un entier positif. La classe des *graphes de largeur NLC-k* est définie récursivement avec les opérations suivantes :

– Pour tout $i \in \{1, \ldots, k\}$, $\bullet_{(i)}$ est dans NLC-k, où $\bullet_{(i)}$ est le graphe à un sommet étiqueté i.
– Soient $G_1 = (V_1, E_1, l_1)$ et $G_2 = (V_2, E_2, l_2)$ deux graphes NLC-k et soit $S \subseteq \{1, \ldots, k\}^2$. Alors $G_1 \times_S G_2$ est NLC-k, où $G_1 \times_S G_2 = (V, E, l)$avec $V = V_1 \cup V_2$,

$$E = E_1 \cup E_2 \cup \{\{u, v\} : (u, v) \in V_1 \times V_2 \text{ et } (l_1(u), l_2(v)) \in S\}$$

$$\text{et pour tout } u \in V,\ l(u) = \begin{cases} l_1(u) \text{ si } u \in V_1 \\ l_2(u) \text{ si } u \in V_2. \end{cases}$$

– Soient $R : \{1, \ldots, k\} \to \{1, \ldots, k\}$ et $G = (V, E, l)$ un graphe k-étiqueté. Alors $\rho_R(G)$ est dans NLC-k, où $\rho_R(G) = (V, E, l')$ tel que $l'(u) = R(l(u))$ pour tout $u \in V$.

Un graphe est NLC-k s'il existe un k-étiquetage de G tel que (G, l) est dans NLC-k. Un graphe k-étiqueté est *NLC-k ρ-free* s'il peut être obtenu sans faire usage de l'opération ρ_R.

2.0.1. *Modules et décomposition modulaire :*

Nous rappelons que $\mathscr{M}(G)$ représente les modules de G et $\mathscr{M}'(G)$ représente les modules forts de G. Et soit $\mathcal{P}(G) = \{M_1, \ldots, M_k\}$ les éléments maximaux (vis à vis de l'inclusion) de $\mathscr{M}'(G) \setminus \{V\}$.

$\mathcal{P}(G)$ est une partition de V, et G peut être décomposé en $G[M_1], \ldots, G[M_k]$, où $\mathcal{P}(G) = \{M_1, \ldots, M_k\}$. Le *graphe caractéristique* G^* d'un G est le graphe ayant pour sommets $\mathcal{P}(G)$ et deux $P, P' \in \mathcal{P}(G)$ sont adjacents s'il y a une arête entre P et P' dans G.

Pour $M \in \mathscr{M}'(G)$, soit $G_M = G[M]$ et G^*_M son graphe caractéristique.

LEMME 5.1. *Johansson (2000) Soit G un graphe. G est NLC-k si et seulement si tout graphe caractéristique dans la décomposition modulaire de G est NLC-k.*

De plus, une expression NLC-k de G peut facilement être construite à partir de la décomposition modulaire et des expressions NLC-k des graphes premiers. Sur les graphes premiers, la reconnaissance de NLC-2 est plus facile.

LEMME 5.2. *Johansson (2000) Soit G un graphe premier. Alors G est de NLC-2 si et seulement si il existe un 2-étiquetage l tel que (G, l) est NLC-2 ρ-free.*

3. Reconnaissance des graphes NLC-2

3.1. NLC-2 ρ-free Décomposition canonique.

Dans cette section, $G = (V, E, l)$ est un graphe 2-étiqueté tel que chaque module unicolore (*i.e.* un module M tel que $\forall v, v' \in M$, $l(v) = l(v')$) a une taille égale à 1. Un couple (X, Y) est un *coupe-x* si $X \cup Y = V$, $X \cap Y = \varnothing$, $X \neq \varnothing$ et $Y \neq \varnothing$. Soit $S \subseteq \{1, 2\} \times \{1, 2\}$. Une coupe-x (X, Y) est une *S-coupe* de G si pour tout $u \in X$ et $v \in Y$, alors $\{u, v\} \in E$ si et seulement si $(l(u), l(v)) \in S$. Pour $S \subseteq \{1, 2\} \times \{1, 2\}$ soit $\mathcal{F}_S(G)$ l'ensemble des S-coupes de G.

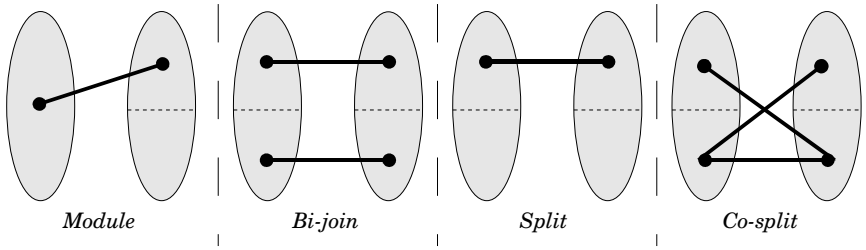


Figure 5.1. Un module, un bi-join, une coupe et une co-coupe

DÉFINITION 5.3 (Symétrie). $S \in \{1,2\} \times \{1,2\}$ est dit *symétrique* si $(1,2) \in S \iff (2,1) \in S$, sinon nous disons que S n'est pas *symétrique*.

DÉFINITION 5.4 (Propriété dégénérée). Une famille $\mathcal{F}$ de coupes-x a la *propriété dégénérée* s'il existe une partition $\mathcal{P}$ de V tel que pour tout $\varnothing \subsetneq \mathcal{X} \subsetneq \mathcal{P}$, $(\bigcup_{X\in\mathcal{X}} X, \bigcup_{Y\in\mathcal{P}\setminus\mathcal{X}} Y)$ est dans $\mathcal{F}$, et il n'y pas d'autres coupe-x $\mathcal{F}$.

LEMME 5.5. *Pour tout $S \subseteq \{1,2\} \times \{1,2\}$ symétrique, $\mathcal{F}_S(G)$ a la propriété dégénérée.*

DÉMONSTRATION. La famille $\mathcal{F}_{\{\}}(G)$ a la propriété dégénérée car comme (X,Y) est une $\{\}$-cut si et seulement sil n'y a aucune arête entre X et Y ($\mathcal{P}$ est exactement l'ensemble des composantes connexes). Pour $W \subseteq V$, soit $G|W = (V, E\Delta W^2, l)$. Pour $i \in \{1,2\}$, soit $V_i = \{v \in V : l(v) = i\}$. Soit $G_1 = G|V_1$, $G_2 = G|V_2$ et $G_{12} = (G|V_1)|V_2$.

- $\mathcal{F}_{\{(1,1)\}}(G) = \mathcal{F}_{\{\}}(G_1)$, $\mathcal{F}_{\{(2,2)\}}(G) = \mathcal{F}_{\{\}}(G_2)$, $\mathcal{F}_{\{(1,1),(2,2)\}}(G) = \mathcal{F}_{\{\}}(G_{12})$,
- $\mathcal{F}_{\{(1,1),(1,2),(2,1),(2,2)\}}(G) = \mathcal{F}_{\{\}}(\overline{G})$, $\mathcal{F}_{\{(1,2),(2,1),(2,2)\}}(G) = \mathcal{F}_{\{\}}(\overline{G_1})$,
  $\mathcal{F}_{\{(1,1),(1,2),(2,1)\}}(G) = \mathcal{F}_{\{\}}(\overline{G_2})$, $\mathcal{F}_{\{(1,2),(2,1)\}}(G) = \mathcal{F}_{\{\}}(\overline{G_{12}})$.

Donc pour tout $S \subseteq \{1,2\} \times \{1,2\}$ symétrique , $\mathcal{F}_S(G)$ a la propriété dégénérée. □

DÉFINITION 5.6 (Propriété Linéaire). Une famille $\mathcal{F}$ de coupes-x a la *propriété linéaire* si pour tout (X,Y) et (X',Y') dans $\mathcal{F}$, soit $X \subseteq X'$ soit $X' \subseteq X$.

LEMME 5.7. *Pour tout $S \subseteq \{1,2\} \times \{1,2\}$ non-symétrique , $\mathcal{F}_S(G)$ a la propriété linéaire.*

DÉMONSTRATION. Cas où $S = \{(1,2)\}$: supposons que $X \setminus X'$ et $X' \setminus X$ sont tous les deux non vides. Alors si $u \in X \setminus X'$ est étiqueté 1 et $v \in X' \setminus X$ est étiqueté 2, u et v doivent être adjacents et non-adjacents, contradiction. Par conséquent, $X \setminus X'$ et $X' \setminus X$ sont unicolores. Supposons maintenant, sans perte de généralité, que tous les sommets dans $X\Delta X'$ sont étiquetés 1. Alors $X\Delta X'$ est adjacent à tous les sommets étiquetés 2 dans $Y \cap Y'$ et non adjacents à tous les sommets 1 dans $Y \cap Y'$. De plus, $X\Delta X'$ est non adjacent à tous les sommets dans $X \cap X'$. Donc $X\Delta X'$ est un module unicolore, et $|X\Delta X'| \geq 2$. Contradiction. Pour les autres S non-symétriques, nous nous ramenons au cas $\{(1,2)\}$ comme dans la preuve du lemme 5.5. □

Pour $S \subseteq \{1,2\} \times \{1,2\}$, $\mathcal{P}_S(G)$ dénote l'unique partition de V telle que

(1) pour tout $(X,Y) \in \mathcal{F}_S(G)$ et $P \in \mathcal{P}_S(G)$, $P \subseteq X$ ou $P \subseteq Y$, et

(2) pour tout $P, P' \in \mathcal{P}$, $P \neq P'$, il existe $(X,Y) \in \mathcal{F}_S(G)$ tels que $P \subseteq X$ et $P' \subseteq Y$, ou $P \subseteq Y$ et $P' \subseteq X$.

Pour un $S \in \{1,2\} \times \{1,2\}$ non-symétrique , $\mathcal{P}'_S(G) = (P_1, \ldots, P_k)$ représente l'unique ordre des éléments dans $\mathcal{P}_S(G)$ tel que pour tout $(X,Y) \in \mathcal{F}_S(G)$, il existe l tel que $X = \cup_{i \in \{1,\ldots,l\}} P_i$.

LEMME 5.8. *Si G est dans NLC-2 ρ-free, alors il existe $S \subseteq \{1,2\} \times \{1,2\}$ tel que $\mathcal{F}_S(G)$ est non vide.*

DÉMONSTRATION. Si G est NLC-2 ρ-free, alors il existe un $S \subseteq \{1,2\} \times \{1,2\}$, et deux graphes G_1 et G_2 tels que $G = G_1 \times_S G_2$. Par conséquent $(V(G_1), V(G_2)) \in \mathcal{F}_S(G)$ et $\mathcal{F}_S(G)$ est non vide. □

LEMME 5.9. *Soit $G = (V, E, l)$ un graphe 2-étiqueté et soit $S \subseteq \{1,2\} \times \{1,2\}$. Si G est NLC-2 ρ-free et n'a pas de module unicolore non-trivial, alors pour tout $P \in \mathcal{P}_S(G)$, $G[P]$ n'a pas de module unicolore non-trivial.*

DÉMONSTRATION. Si M est un module unicolore de $G[P]$, alors M est un module unicolore de G. Contradiction. □

LEMME 5.10. *Soit $G = (V, E, l)$ un graphe 2-étiqueté et soit $S \subseteq \{1,2\} \times \{1,2\}$. Alors G est NLC-2 ρ-free si et seulement si pour tout $P \in \mathcal{P}_S(G)$, $G[P]$ est NLC-2 ρ-free.*

DÉMONSTRATION. Le "seulement si" est immédiat. Supposons maintenant que pour tout $P \in \mathcal{P}_S(G)$, $G[P]$ est NLC-2 ρ-free. Si S est symétrique, soit $\mathcal{P}_S(G) = \{P_1, \ldots, P_{|\mathcal{P}_S(G)|}\}$. Alors $G = ((G[P_1] \times_S G[P_2]) \times_S \ldots \times_S G[P_{|\mathcal{P}_S(G)|}]$, et G est NLC-2 ρ-free. Autrement, si S est non-symétrique, soit $\mathcal{P}'_S(G) = (P_1, \ldots, P_{|\mathcal{P}_S(G)|})$. Alors $G = ((G[P_1] \times_S G[P_2]) \times_S \ldots \times_S G[P_{|\mathcal{P}_S(G)|}]$, et G est NLC-2 ρ-free. □

L' *arbre de décomposition NLC-2 ρ-free* d'un graphe 2-étiqueté G est un arbre enraciné tel que les feuilles sont les sommets de G, et les noeuds internes sont étiquetés $\times_S$, avec $S \subseteq \{1,2\} \times \{1,2\}$. Un noeud interne est `dégénéré` si S est symétrique, et `linéaire` si S est non-symétrique. Avec les lemmes 5.8,5.9 et 5.10, G est NLC-2 ρ-free si et seulement si il a un arbre de décomposition NLC-2 ρ-free. Cet arbre de décomposition est unique. Mais, nous pouvons définir un *arbre de décomposition canonique* si nous fixons un ordre total sur les sous-ensembles de $\{1,2\} \times \{1,2\}$ (par exemple l'ordre lexicographique). Si deux graphes sont isomorphes, alors ils ont le même arbre de décomposition canonique. L'algorithme 7 calcule l'arbre de décomposition canonique d'un graphe 2-étiqueté premier, ou échoue si G n'est pas NLC-2 ρ-free.

Entrées : Un graphe 2-étiqueté $G = (V, E, l)$
Sorties : Un arbre de décomposition NLC-2 ρ-free, ou échoue G n'est pas NLC-2 ρ-free

si $|V| = 1$ **alors**
  retourner *la feuille* $\bullet_{(l(v))}$ (où $V = \{v\}$)
Soit $\mathcal{S}$ l'ensemble des sous-ensembles de $\{1, 2\} \times \{1, 2\}$ et σ l'ordre lexicographique de $\mathcal{S}$
pour chaque $S \in \mathcal{S}$ *vis à vis de* σ **faire**
  Calculer $\mathcal{P}_S(G)$, et $\mathcal{P}'_S(G)$ si S est non-symétrique (voir l'algorithme 8)
  si $|\mathcal{P}_S(G)| > 1$ **alors**
    Créer un nouveau noeud $\times_S$ β
    pour chaque $P \in \mathcal{P}_S(G)$ *(vis à vis de* $\mathcal{P}'_S(G)$ *si* S *est non-symétrique)* **faire**
      l'arbre de décomposition NLC-2 ρ-free de $G[P]$ devient un enfant de β.
    retourner *L'arbre enraciné en* β
échoue avec *pas NLC-2 ρ-free*

Algorithme 7 : Calcul de l'arbre canonique NLC-2 ρ-free

L'algorithme 8 calcule $\mathcal{P}_S$ et $\mathcal{P}'_S$ pour un graphe premier 2-étiqueté G et $S \subseteq \{1, 2\} \times \{1, 2\}$ en temps linéaire. Nous avons besoin de quelques définitions supplémentaires pour cet algorithme et sa preuve de correction. Un *graphe biparti* est un triplet (X, Y, E) tel $E \subseteq X \times Y$. Le *bi-complément* d'un graphe biparti (X, Y, E) est le graphe biparti $(X, Y, (X \times Y) \setminus E)$. Un *trigraphe biparti (BT)* est un graphe biparti avec deux types d'arêtes : les arêtes *join* et les arêtes *mixtes* . C'est noté $\mathcal{B} = (X, Y, E_j, E_m)$ où E_j est l'ensemble des arêtes *join*, et E_m est l'ensemble des arêtes *mixtes*. Un *BT-module* dans un trigraphe biparti est un sous-ensemble $M \subseteq X$ ou $M \subseteq Y$ tel que M est un module dans (X, Y, E_j) et où il n'y a pas d'arêtes *mixtes* entre M et $(X \cup Y) \setminus M$. Pour $v \in X \cup Y$, soit $N_j(v) = \{u \in X \cup Y : \{u, v\} \in E_j\}$ et $N_m(v) = \{u \in X \cup Y : \{u, v\} \in E_m\}$. Soit $d_j(v) = |N_j(v)|$ et $d_m(v) = |N_m(v)|$. Un *semi-join* dans un trigraphe biparti (X, Y, E_j, E_m) est un coupe-x (A, B) de $X \cup Y$, tel qu'il n'y ait pas d'arêtes entre $A \cap Y$ et $B \cap X$, et il y a seulement des arêtes de type *join* entre $A \cap X$ et $B \cap Y$.

Dans l'algorithme 8, $\mathcal{B}$ est obtenu à partir du graphe G. Les sommets de X correspondent aux sous-ensembles de sommets étiquetés 1 dans G, et les sommets Y correspondent aux sous-ensembles de sommets étiquetés 2. Il y a une arête *join* entre M et M' dans $\mathcal{B}$ si M ① M' dans G, et il y a une arête *mixe* entre $M \in X$ et $M' \in Y$ dans $\mathcal{B}$ s'il y a au moins une arête et une non-arête entre M et M' dans G. Un tel graphe $\mathcal{B}$ peut facilement être construit en temps linéaire à partir d'un graphe donné G. Il suffit de considérer une liste et un tableau borné par nombre de composantes dans G de la même couleur.

Les lemmes suivants sont des observations proches de celles présentées dans les travaux de Fouquet et al. (1999), à la différence qu'ici l'objet étudié considéré est un trigraphe et non plus un graphe biparti.

LEMME 5.11. *Soit $G = (X, Y, E_j, E_m)$ un trigraphe biparti tel que tout BT-module a pour cardinalité 1. Soit $(x_1, \ldots, x_{|X|})$ est X trié par $(d_j(x), d_m(x))$ selon l'ordre lexicographique décroissant. Si (A, B) est un semi-join de G, alors il existe un $k \in \{0, \ldots, |X|\}$ tel que $A \cap X = \{x_1, \ldots, x_k\}$.*

DÉMONSTRATION. pour tout $v \in A \cap X$, $d_j(v) \geq |B \cap Y|$, et pour tout $v \in B \cap X$, $d_j(v) \leq |B \cap Y|$. De plus, s'il existe un $v \in B \cap X$ avec $d_j(v) = |B \cap Y|$, alors $d_m(v) = 0$. Soit $C = \{v \in X : d_j(v) = |B \cap Y| \text{ et } d_m(v) = 0\}$. Alors C est un BT-module de G, et donc $|C| \leq 1$. Tout sommet dans $A \cap X \setminus C$ sont avant tout sommet dans $B \cap X \setminus C$ selon l'ordre. Qui plus est, si $|C| > 0$, alors les sommets $A \cap X \setminus C$ sont avant les sommets dans C, et les sommets dans $B \cap X \setminus C$ sont après le sommet dans C selon l'ordre. □

Entrées : Un graphe 2-étiqueté G, et $S \subseteq \{1, 2\} \times \{1, 2\}$
Sorties : $\mathcal{P}_S$ si S est symétrique, $\mathcal{P}'_S$ si S n'est pas symétrique

$V_i \leftarrow \{v : v \in V \text{ et } l(v) = i\}$
si $(1, 1) \in S$ **alors**
  $\mathcal{C}_1 \leftarrow$ composantes co-connexes de $G[V_1]$
sinon
  $\mathcal{C}_1 \leftarrow$ composantes connexes de $G[V_1]$
si $(2, 2) \in S$ **alors**
  $\mathcal{C}_2 \leftarrow$ composantes co-connexes de $G[V_2]$
sinon
  $\mathcal{C}_2 \leftarrow$ composantes connexes de $G[V_2]$
$\mathcal{B} = (\mathcal{C}_1, \mathcal{C}_2, E_j, E_m) \leftarrow$ Le trigraphe biparti entre les éléments de $\mathcal{C}_1$ et $\mathcal{C}_2$;
si $S \cap \{(1, 2), (2, 1)\} = \varnothing$ **alors**
  retourner *composantes connexes de* $(\mathcal{C}_1, \mathcal{C}_2, E_j \cup E_m)$
sinon si $S \cap \{(1, 2), (2, 1)\} = \{(1, 2), (2, 1)\}$ **alors**
  return *composantes connexes du bi-complémentaire de* $(\mathcal{C}_1, \mathcal{C}_2, E_j)$
sinon
  chercher tous les semi-joins de $\mathcal{B}$ (en utilisant les lemmes 5.11 et 5.12)

Algorithme 8 : Calcul de $\mathcal{P}_S$ et $\mathcal{P}'_S$

LEMME 5.12. *Soit $k \in \{0, \ldots, |X|\}$ et $k' \in \{0, \ldots, |Y|\}$. Alors $(A, (X \cup Y) \setminus A)$, où $A = \{x_1, \ldots, x_k, y_1, \ldots, y_{k'}\}$, est un semi-join de G si et seulement si $\sum_{i=1}^{k} d_j(x_i) - \sum_{i=1}^{k'} d_j(y_i) = k \times (|Y| - k')$ et $\sum_{i=1}^{k} d_m(x_i) - \sum_{i=1}^{k'} d_m(y_i) = 0$.*

DÉMONSTRATION. Le "si " est vrai par définition. Considérons maintenant le "Seulement si". Supposons que la condition du degré est vérifiée, mais que (A, B) ne soit par un semi-join. Nous noterons a comme étant le nombre d'arêtes de type *join* entre $A \cap X$ et $B \cap Y$, et b le nombre d'arêtes de type join entre $A \cap X$ et $A \cap Y$, et c le nombre d'arêtes mixtes entre $A \cap X$ et $A \cap Y$. Remarquons que $a \leq k(|Y| - k')$, et $b \leq \sum_{i=1}^{k'} d_j(y_i)$, par conséquent $a \geq k(|Y| - k')$. Nous avons donc $a = k(|Y| - k')$, et $\sum_{i=1}^{k'} d_j(y_i) - b = 0$. En d'autres mots, il y a seulement des arêtes de types *join* $A \cap X$ et $B \cap Y$, et il n'y a aucune arête de type join entre $A \cap Y$ et $B \cap X$. Maintenant comme il y a seulement des arêtes *join* entre $A \cap X$ et $B \cap Y$, $c = \sum_{i=1}^{k} d_m(x_i) = \sum_{i=1}^{k'} d_m(y_i)$, et donc il n'y a pas d'arêtes mixtes entre $A \cap Y$ et $B \cap X$. □

THÉORÈME 5.13. *L'algorithme 8 est correct et se termine en temps linéaire.*

DÉMONSTRATION.

Correction : Supposons que (A, B) soit une S-coupe. Si $(1, 1) \notin S$, alors il n'y a pas d'arêtes entre $A \cap V_1$ et $B \cap V_1$, donc (A, B) ne peut pas être une composante de coupe $\mathcal{C}_1$ (et de la même manière pour $(1, 1) \in S$, et pour $\mathcal{C}_2$). Nous travaillons maintenant sur le trigraphe biparti $\mathcal{B} = (\mathcal{C}_1, \mathcal{C}_2, E_j, E_m)$. Si $S \cap \{(1, 2), (2, 1)\} = \varnothing$, alors les S-coupes correspondent exactement aux composantes connexes de $\mathcal{B}$, et si $S \cap \{(1, 2), (2, 1)\} = \{(1, 2), (2, 1)\}$ alors les S-coupes correspondent exactement aux composantes connexes du trigraphe biparti de $\overline{G}$, qui est $(\mathcal{C}_1, \mathcal{C}_2, (\mathcal{C}_1 \times \mathcal{C}_2) \setminus (E_j \cup E_m), E_m)$. Finalement, si S est non-symétrique, les S-coupes correspondent aux semi-joins de $\mathcal{B}$.

Complexité : Il est de notoriété publique que l'on peut effectuer un parcours en largeur sur un graphe ou sur son complément en temps linéaire Habib et al. (1999) ; Dahlhaus et al. (2002). Les instructions lignes [2-8,12] peuvent être faites avec un BFS sur le graphe ou son complément. Il est facile de voir que nous pouvons faire un parcours en largeur sur le bi-complément en temps linéaire (comme un BFS sur le graphe complémentaire, avec deux listes de sommets pour X et Y), donc l'instruction ligne 14 peut être effectuée en temps linéaire. Finalement, les opérations ligne 15 sont effectuées en temps linéaire. □

Ces résultats peuvent être résumés par le théorème suivant :

THÉORÈME 5.14. *L'algorithme 7 calcule l'arbre de décomposition canonique NLC-2 ρ-free d'un graphe 2-étiqueté en temps $O(nm)$.*

3.2. Décomposition NLC-2 d'un graphe premier. Dans cette section, G est un graphe non étiqueté et premier pour la décomposition modulaire avec $|V| \geq 3$.

DÉFINITION 5.15 (2-*bimodule*). Une bipartition $\{X, Y\}$ de V est un 2-*bimodule* si X peut être partitionné en X_1 et X_2, et Y en Y_1 et Y_2 tels que pour tout $(i, j) \in \{1, 2\} \times \{1, 2\}$, alors soit X_i ⓪ Y_j soit X_i ① Y_j. Il est facile de voir que si $\{X, Y\}$ est un 2-bimodule si

et seulement si $\{X, Y\}$ est un split, un co-split ou un bi-join. De plus, si $\min(|X|, |Y|) > 1$ alors $\{X, Y\}$ ne peuvent pas être à la fois split,bi-join,etc... car G est premier.

Soit $l : V \to \{1, 2\}$ un 2-étiquetage. Alors $s(l)$ dénote le 2-étiquetage sur V tel que pour tout $v \in V$, $s(l)(v) = 1$ si et seulement si $l(v) = 2$.

DÉFINITION 5.16 (étiquetage induit par un 2-bimodule). Soit $\{X, Y\}$ un 2-bimodule. Nous définissons *l'étiquetage* $l : V \to \{1, 2\}$ *de* G *induit par* $\{X, Y\}$. Si $|X| = |Y| = 1$, alors $l(x) = 1$ et $l(y) = 2$, où $X = \{x\}$ et $Y = \{y\}$. Si $|X| = 1$, alors $l(v) = 1$ si et seulement si $v \in N[x]$. De manière similaire $|Y| = 1$, alors $l(v) = 1$ si et seulement si $v \in N[y]$. Supposons maintenant que $\min(|X|, |Y|) > 1$. si $\{X, Y\}$ est un split, alors l'ensemble des sommets dans X avec un voisin dans Y et l'ensemble des sommets dans Y avec un voisin dans X est étiqueté 1, les autres sommets sont étiquetés 2. Si $\{X, Y\}$ est un co-split, alors l'étiquetage de G induit par $\{X, Y\}$ est un étiquetage de $\overline{G}$ induit par la coupe $\{X, Y\}$. Finalement si $\{X, Y\}$ est un bi-join, l est tel que $\{v \in X : l(v) = 1\}$ est un join avec $\{v \in Y : l(v) = 1\}$ et $\{v \in X : l(v) = 2\}$ est un join avec $\{v \in Y : l(v) = 2\}$. Remarquons que si $\{X, Y\}$ est un bi-join, alors deux étiquetages sont possibles l_1 et l_2, avec $l_1 = s(l_2)$. Si $\{X, Y\}$ est un 2-bimodule de G et l un étiquetage induit par $\{X, Y\}$, alors chaque module unicolore a une taille 1 (comme G est premier et $|V| \geq 3$).

DÉFINITION 5.17 (Bon 2-bimodule). Un 2-bimodule $\{X, Y\}$ est *bon* si le graphe G avec l'étiquetage induit par $\{X, Y\}$ est NLC-2 ρ-free. La proposition suivante suit immédiatement du lemme 5.2.

PROPOSITION 5.18. *G est NLC-2 si et seulement si G a un bon 2-bimodule.*

LEMME 5.19. *Si G a un bon 2-bimodule $\{X, Y\}$ qui est un split, alors G a un bon 2-bimodule qui est un split fort.*

DÉMONSTRATION. Il y a un noeud α dans l'arbre de décomposition en split et nous avons $\varnothing \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$ tel que $\{X, Y\} = \{\cup_{i \in I} C^i_\alpha, \cup_{i \notin I} C^i_\alpha\}$. Soit $l : V \to \{1, 2\}$ l'étiquetage de G induit par $\{X, Y\}$. Pour tout $i \in \{1, \ldots, d(\alpha)\}$, $(G[C^i_\alpha], l|_{C^i_\alpha})$ est NLC-2 ρ-free (où $l|_W$ est la fonction l restreinte à W).

Soit l' un 2-étiquetage de V tel que pour tout i, et $v \in C^i_\alpha$, $l(v) = 1$ si et seulement si v a un voisin à l'extérieur de C^i_α. Pour tout i, soit $l|_{C^i_\alpha} = l'|_{C^i_\alpha}$, ou $\forall v \in C^i_\alpha$, $l(v) = 2$. Alors pour tout i, $(G[C^i_\alpha], l'|_{C^i_\alpha})$ est NLC-2 ρ-free, et donc (G, l') est NLC-2 ρ-free. Comme il y a un sommet dominant dans le graphe caractéristique de α, il existe j tel que l'étiquetage induit par le split fort $\{C^j_\alpha, V \setminus C^j_\alpha\}$ est l'. Donc le split fort $\{C^j_\alpha, V \setminus C^j_\alpha\}$ est bon. □

Les lemmes précédents sur $\overline{G}$ disent que si G a un bon 2-bimodule $\{X, Y\}$ qui est un co-split, alors G a un bon 2-bimodule qui est un co-split fort. Le lemme suivant est similaire au lemme 5.19.

LEMME 5.20. *Si G a un bon 2-bimodule $\{X, Y\}$ qui est un bi-join, alors G a un bon 2-bimodule qui est bi-join fort.*

Entrées : Un graphe G
Sorties : Oui si et seulement si G est de NLC-2
$\mathcal{S} \leftarrow$ l'ensembles des coupes fortes, co-coupes fortes et des bi-joins forts de G
pour chaque $\{X, Y\} \in \mathcal{S}$ **faire**
  $l \leftarrow$ l'étiquetage de G induit par $\{X, Y\}$
  si $(G[X], G[Y], l)$ *est NLC-2 ρ-free* **alors**
    retourner *Oui*
retourner *Non*

Algorithme 9 : Algorithme de reconnaissance des graphes NLC-2 premiers.

THÉORÈME 5.21. *L'algorithme 9 reconnaît les graphes NLC-2 premiers, et sa complexité est en temps $O(n^2.m)$.*

DÉMONSTRATION. De manière triviale, si l'algorithme renvoie "oui", alors le graphe G est NLC-2. D'autre part, la proposition 5.18, et les lemmes 5.19 et 5.20, si G est NLC-2, alors il comporte un bon 2-bimodule et l'algorithme renvoie "oui".

L'ensemble $\mathcal{S}$ peut être calculé en utilisant les algorithmes pour calculer la décomposition en split Dahlhaus (2000) sur G et sur $\overline{G}$, et la décomposition en bi-join sur G. Remarquons simplement que si l'usage d'un algorithme linéaire de décomposition en split n'est pas requis, il est possible d'utiliser des algorithmes plus simples que celui présenté par Dahlhaus (2000). En effet Cunningham (1982) ; Gabor et al. (1989) proposent des algorithmes en $O(n^2.m)$ ou encore Ma et Spinrad (1994) proposent un algorithme en $O(n^2)$.

Montgolfier et Rao (2005a,b) montrent que la décomposition en bi-join peut être calculée en temps linéaire, en utilisant une réduction au problème de décomposition modulaire. Nous pourrons utiliser au choix, si l'on souhaite un algorithme linéaire, McConnell et Spinrad (1999) ou de manière plus simple Tedder et al. (2008).

Il existe toutefois des algorithmes plus simples, du moment que la complexité attendue n'est pas linéaire. L'ensemble $\mathcal{S}$ a $O(n)$ éléments. Tester si un 2-bimodule est bon, prend un temps $O(n.m)$ en utilisant l'algorithme 7. Par suite de quoi le temps d'exécution de l'algorithme est borné par $O(n^2m)$. □

3.3. Décomposition NLC-2. En utilisant le lemme 5.1, la décomposition modulaire et l'algorithme 9, nous obtenons le théorème suivant ;

THÉORÈME 5.22. *Les graphes NLC-2 peuvent être reconnus en temps $O(n^2m)$, et une expression NLC-2 peut être générée dans la même complexité.*

4. Isomorphisme de graphes sur les graphe NLC-2

4.1. Isomorphisme de graphes sur les graphes premiers NLC-2 ρ-free.

PROPOSITION 5.23. *Considérons un $S \in \{1,2\} \times \{1,2\}$ symétrique. Deux graphes G et H sont isomorphes si et seulement si il y a une bijection π entre $\mathcal{P}_S(G)$ et $\mathcal{P}_S(H)$ tel que pour tout $P \in \mathcal{P}_S(G)$, $G[P]$ est isomorphe à $H[\pi(P)]$.*

PROPOSITION 5.24. *Soit $S \in \{1,2\}\times\{1,2\}$ non-symétrique et soit G et H deux graphes. Soit $\mathcal{P}'_S(G) = (P_1, \ldots, P_k)$ et $\mathcal{P}'_S(H) = (P'_1, \ldots, P'_{k'})$ alors G et H sont isomorphes si et seulement si $k = k'$ et pour tout $i \in \{1, \ldots, k\}$, $G[P_i]$ est isomorphe à $H[P'_i]$.*

Ces deux propositions sont des conséquences directes des propriétés linéaires et dégénérées des S-coupes. Alors deux graphes 2-étiquetés NLC-2 ρ-free G et H sont isomorphes si et seulement si il y a un isomorphisme entre leurs arbres de décomposition NLC-2 ρ-free qui respecte l'ordre des enfants des noeuds `linéaires`. Cet isomorphisme peut être testé en temps linéaire, par conséquent l'isomorphisme de graphes NLC-2 ρ-free peut être effectué en temps $O(n.m)$.

Entrées : Deux graphes premiers NLC-2 G et H
Sorties : Oui si $G \simeq H$, Non sinon
$\mathcal{S} \leftarrow$ l'ensemble des splits forts, co-splits et bi-joins de G
$\mathcal{S}' \leftarrow$ l'ensemble des splits forts, co-splits et bi-joins de H
si *il n'y a pas de bon 2-bimodule dans* $\mathcal{S}$ **alors**
  Echec : "G n'est pas NLC-2"
$\{X, Y\} \leftarrow$ un bon 2-bimodule de $\mathcal{S}$
$l \leftarrow$ l'étiquetage de G induit par $\{X, Y\}$
pour chaque $\{X', Y'\} \in \mathcal{S}'$ *tel que* $\{X', Y'\}$ *est bon* **faire**
  $l' \leftarrow$ l'étiquetage de H induit par $\{X', Y'\}$
  si $|X| > 1$ **et** $|Y| > 1$ **et** $\{X, Y\}$ *est un bi-join* **alors**
    si $(G, l) \simeq (H, l')$ **ou** $(G, l) \simeq (H, s(l'))$ **alors**
      return *Oui*
  sinon
    si $(G, l) \simeq (H, l')$ **alors**
      return *Oui*
retourner *Non*

Algorithme 10 : Isomorphisme pour les graphes NLC-2 premiers

4.2. Isomorphisme des graphe NLC-2 premiers.

THÉORÈME 5.25. *L'algorithme 10 teste l'isomorphisme entre deux graphes NLC 2 premiers en temps $O(n^2m)$.*

DÉMONSTRATION. Si l'algorithme répond "oui", alors de manière évidente $G \simeq H$. Supposons maintenant que $G \simeq H$ et soit $\pi : V(G) \to V(H)$ une bijection telle que $\{u, v\} \in E(G)$ si et seulement si $(\pi(u), \pi(v)) \in E(H)$. Alors $\{X', Y'\}$ avec $X' = \pi(X)$ et $Y' = \pi(Y)$ est un bon 2-bimodule si H. Si $\min(|X|, |Y|) > 1$ et $\{X', Y'\}$ est un bi-join, alors par définition il existe deux étiquetages induit par $\{X, Y\}$, et $(G, l) \simeq (H, l')$ ou $(G, l) \simeq (H, s(l'))$. Sinon l'étiquetage est unique et $(G, l) \simeq (H, l')$.

Les ensembles $\mathcal{S}$ et $\mathcal{S}'$ peuvent être calculés en temps $O(n^2)$ en utilisant les algorithmes linéaire pour calculer la décomposition en split sur G et $\overline{G}$, et la décomposition en sur G. Les ensembles $\mathcal{S}$ et $\mathcal{S}'$ ont $O(n)$ éléments. Tester si un 2-bimodule est bon prend un temps $O(nm)$ en utilisant l'algorithme 7, et teste si deux graphes premiers 2-étiquetés sont isomorphes prend un temps $O(nm)$. Le temps total d'exécution est $O(n^2 m)$. □

4.3. Isomorphisme des graphes NLC-2. Il est facile de montrer que l'isomorphisme de graphe sur les graphes NLC-2 premiers avec un étiquetage supplémentaire dans $\{1, \ldots, q\}$ peut être fait en temps $O(n^2 m)$. Pour cela nous ajoutons l'étiquette supplémentaire de v à la feuille correspondante à v dans l'arbre de décomposition NLC-2 ρ-free.

Nous montrons que nous pouvons tester l'isomorphisme de graphes NLC 2 en temps $O(n^2.m)$, en utilisant la décomposition modulaire et l'algorithme 10.

Soient $\mathscr{M}'(G)$ et $\mathscr{M}'(H)$ la décomposition modulaire de G et de H. Pour $M \in \mathscr{M}'(G)$, soit G_M étant $G[M]$, et pour $M \in \mathscr{M}'(H)$, soit H_M étant $H[M]$. Soit G^*_M le graphe caractéristique de G_M (remarquons que $|V(G^*_M)|$ est le nombre d'enfants de M dans l'arbre de décomposition modulaire). Soit $\mathscr{M}'_{(i,*)} = \{M \in \mathscr{M}'(G) \cup \mathscr{M}'(H) : |M| = i\}$, soit $\mathscr{M}'_{(*,j)} = \{M \in \mathscr{M}'(G) \cup \mathscr{M}'(H) : |V(G^*_M)| = j\}$ et soit $\mathscr{M}'_{(i,j)} = \mathscr{M}'_{(i,*)} \cap \mathscr{M}'_{(*,j)}$. Remarquons que $\sum_{j=1}^{n} (|\mathscr{M}'_{(*,j)}| \times j)$ est le nombre de sommets dans G plus le nombre d'arêtes dans l'arbre de décomposition modulaire, et donc comporte au plus $3n - 2$ arêtes.

Entrées : Deux graphes NLC-2 G et H
Sorties : Oui si $G \simeq H$, Non sinon
pour chaque $M \in \mathscr{M}'(G) \cup \mathscr{M}'(H)$ *tels que* $|M| = 1$ **faire**
  $l(M) \leftarrow 1$
pour i **de** 2 **à** n **faire**
  pour j **de** 2 **à** i **faire**
    Calculer la partition $\mathcal{P}$ de $\mathscr{M}'_{(i,j)}$ telle que M et M' sont dans la même classe de $\mathcal{P}$ si et seulement si $(G^*_M, l) \simeq (G^*_{M'}, l)$.
    pour chaque $P \in \mathcal{P}$ **faire**
      $a \leftarrow$ une nouvelle étiquette (un entier $\notin \mathrm{Img}(l)$)
      pour tout $M \in P$, $l(M) \leftarrow a$

Algorithme 11 : Isomorphisme des graphes NLC-2

THÉORÈME 5.26. *L'algorithme 11 teste l'isomorphisme deux graphes NLC 2 en temps* $O(n^2m)$.

DÉMONSTRATION. La correction vient du fait qu'à chaque étape, pour tout $M, M' \in \mathscr{M}'(G) \cup \mathscr{M}'(H)$ tels que $l(M)$ et $l(M')$ sont des ensembles, G_M et $G_{M'}$ sont isomorphes si et seulement si $l(M) = l(M')$. Le temps total de cet algorithme est $f(n,m)$(le "grand O" est omis) : $f(n,m) \leq \sum_i \sum_j \left(j^2 m |\mathscr{M}'_{(i,j)}|^2\right) \leq m \sum_j \left(j^2 \sum_i \left(|\mathscr{M}'_{(i,j)}|^2\right)\right) \leq m \sum_j \left(j^2 |\mathscr{M}'_{(*,j)}|^2\right)$ $\leq m \sum_j \left(\left(j|\mathscr{M}'_{(*,j)}|\right)^2\right) \leq n^2 m.$

$$\begin{aligned} f(n,m) &\leq \sum_i \sum_j \left(j^2 m |\mathscr{M}'_{(i,j)}|^2\right) \leq m \sum_j \left(j^2 \sum_i \left(|\mathscr{M}'_{(i,j)}|^2\right)\right) \\ &\leq m \sum_j \left(j^2 |\mathscr{M}'_{(*,j)}|^2\right) \leq m \sum_j \left(\left(j|\mathscr{M}'_{(*,j)}|\right)^2\right) \leq n^2 m. \end{aligned}$$

□

Conclusions et perspectives

Ce manuscrit est composé de deux parties pour le moins distinctes. La première partie s'attache à présenter et étudier deux généralisations polynomiales de la décomposition modulaire. La seconde partie, essentiellement algorithmique, présente deux algorithmes. Le premier consiste à calculer les composantes de chevauchement d'une famille de parties d'un ensemble. Le second reconnaît les graphes de largeur NLC 2. Nous présentons maintenant un résumé de nos travaux et les pistes que nous souhaitons poursuivre à l'avenir.

Nous avons présenté au **Chapitre.** 2 une nouvelle structure combinatoire appelée *relations homogènes*. Nous avons introduit cette structure afin de fournir un cadre général pour l'étude de la décomposition modulaire. Pour cela nous utilisons la notion fondamentale de distinction. Nous avons naturellement étudié les propriétés de la décomposition modulaire sur cette structure. Dans un premier temps nous avons montré que la famille des modules a la propriété de former une famille *intersectante* sur les relations homogènes arbitraires. Par la suite, nous nous sommes intéressé aux aspects algorithmiques de cette décomposition. Nous présentons deux algorithmes sur les relations homogènes arbitraires : nous testons en temps linéaire, *i.e.* $O(n^2)$, si une relation homogène est indécomposable, nous présentons ensuite un algorithme en temps $O(n^3)$ pour calculer l'arbre de décomposition modulaire généralisé d'une relation homogène. Par la suite, nous avons étudié des familles particulières de relations homogènes. Il s'agit de relations homogènes équipées de propriétés supplémentaires. En s'aidant de ces nouvelles propriétés, nous avons montré que la famille des modules sur ces relations forme une famille faiblement partitive et nous avons présenté un algorithme linéaire pour calculer l'arbre de décomposition de ces familles. Finalement, nous avons comment utiliser les relations homogènes et le concept de distinction pour introduire des définitions alternatives des modules sur les graphes.

Bien que nous commençons à comprendre un peu la structure des relations homogènes, elles constituent un nouveau domaine de recherche et de nombreux problèmes restent à considérer. En effet, il ne nous apparaît pas comme évident que la complexité de l'algorithme de décomposition modulaire des relations homogènes arbitraire soit optimal. Nous sommes capable de tester la primalité en temps $O(n^2)$ et décomposer seulement en temps $O(n^3)$. Par ailleurs, nous avons donné une caractérisation des relations homogènes des

graphes et des *tournois* mais rien pour le moment concernant les graphes orientés et les graphes dirigés.

Existe-t-il d'autres décompositions de graphes qui peuvent s'exprimer dans ce cadre ? Qu'en est-il pour la décomposition des matroïdes ou plus généralement des hypergraphes ?

En utilisant le cadre fourni pas les relations homogènes, nous avons été amené à changer de point de vue. Au **Chapitre.** 3 nous avons introduit une nouvelle décomposition, la décomposition umodulaire. Cette décomposition se place à l'intérieur de l'ensemble considéré, et non plus à l'extérieur. Nous avons tout d'abord étudié les propriétés des umodules sur les relations homogènes. Nous avons montré que la famille des umodules est close par l'union d'éléments qui se chevauchent. Par la suite nous avons fourni un algorithme polynomial pour trouver la décomposition umodulaire des relations homogènes arbitraires. Nous avons ensuite concentré notre étude sur les umodules des relations homogènes de congruence locale égale à deux. Nous avons montré que la famille des umodules a la propriété de former une famille *crossing*. Nous nous sommes ensuite intéressé aux familles d'umodules auto-complémentée de congruence locale égale à 2. Nous avons montré que la famille des umodules sur ces relations forme une famille bipartitive. Concernant le calcul de la décomposition umodulaire sur ces structures. Nous avons montré comment nous ramener, grâce à une modification locale de la relation homogène, à un calcul de décomposition modulaire sur des *bonnes* relation homogènes. Enfin, nous avons vu à quoi correspond cette décomposition sur les graphes non orientés, et nous avons étudié cette dernière sur les tournois. Sur les tournois nous avons caractérisé les tournois complètement décomposables vis à vis de cette décomposition et présenté deux algorithmes linéaires pour résoudre le problème de l'isomorphisme et le problème du *feedback vertex set* sur les tournois totalement décomposables.

Il s'agit là d'un tout nouveau champ d'étude, et de nombreux problèmes restent à traiter. Le premier est sans doute d'améliorer la complexité de l'algorithme de décomposition des relations homogènes arbitraires, et ainsi descendre en dessous du $O(n^5)$ que nous avons présenté. Il nous semble également intéressant de voir à quoi correspond cette nouvelle décomposition sur les graphes orientés. Une étape intermédiaire a été présenté dans la thèse de Bui-Xuan (2008) avec les *sesqui-modules*. Il serait également intéressant de voir si on peut généraliser l'aproche par modification locale sur les relations homogènes *auto-complémentée* de congruence locale supérieure ou égale à 3. Et enfin, le *Seidel switch* tel qu'utilisé ici semble une opération très prometteuse à étudier. Notamment la notion de *"Seidel minor"*.

Nous avons étudié au **Chapitre.** 4 le problème du calcul des composantes de chevauchements d'une famille de parties. Nous avons, dans un premier temps, présenté l'algorithme de Dahlhaus pour résoudre ce problème, nous avons détaillé son approche et éclairci certaines étapes techniques absentes de son papier. Nous avons ensuite simplifié son algorithme, en se débarrassant d'une procédure algorithmique lourde et complexe à implémenter pour la remplacer par une technique d'affinage de partition. Nous obtenons, tout comme Dahlhaus, un algorithme linéaire pour calculer les composantes de chevauchement. Par la suite nous avons adapté notre algorithme afin d'obtenir comme résultat, un graphe partiel du graphe de chevauchement. L'algorithme ainsi obtenu affiche toujours une complexité linéaire.

Avec le recul, il nous est apparu que le problème des composantes de chevauchements a beaucoup d'applications en algorithmique. Nous avons présenté une version statique du problème, mais il apparaît qu'une version dynamique serait plus approprié pour certains usages. Qu'en est il de la complexité d'ajout ou de retrait d'un élément de la famille, peut-on obtenir la même complexité que la version statique ? Une autre piste qui nous semble importante à développer est le problème des plus petits ancêtre communs. Dans le cadre de l'algorithme de Dahlhaus nous nous sommes débarrassé de cette procédure car peu aisé à implémenter. Cependant cette procédure ne peut pas toujours être évité, et elle s'avère utile voire indispensable dans de nombreux algorithmes. C'est pourquoi il nous semble intéressant de voir s'il est possible d'*implémenter* cette procédure en utilisant seulement des techniques d'affinage de partition. Un autre problème à considérer est lui relié à la propriété du graphe de chevauchement, peut on dire si le graphe a la propriété d'être une chaîne, un arbre, un graphe biparti... Tout cela en temps linéaire ?

Enfin au **Chapitre.** 5 nous avons étudié le problème de reconnaître les graphes de largeur NLC 2. Nous avons présenté un algorithme en temps $O(n^2m)$ améliorant ainsi les résultats précédents. Notre approche utilise des décompositions connexes à la décomposition modulaire. Grâce à ces décompositions et leurs propriétés d'unicité, nous avons pu définir un arbre de décomposition canonique des graphes de largeur NLC 2 et ainsi résoudre le problème de l'isomorphisme sur cette classe en temps polynomial.

Nous avons vu ce qui se passe pour les graphes de largeur NLC 2, une question assez naturelle est : quelle est la situation pour les graphes de largeur NLC 3, où l'opérateur de recoloriage est désormais utile. Et plus généralement pour une constante k fixé est ce que le problème de reconnaître les graphes de NLC-k est un problème FPT ? Une question connexe, est que se passe-t-il concernant la largeur de clique. Pour $k = 3$ il existe un algorithme en $O(n^2m)$ dû à Corneil et al. (2000). Pour k supérieur à 3 et k fixé le problème reste ouvert.

ANNEXE A

Notations

NOTATION A.1 (Voisinage). Soit $G = (V, E)$ un graphe, et x un sommet de V. On notera par $N_X(x)$ voisinage de x dans X, $N_X(x) = \{u : ux \in E \text{ et } u \in X\}$. Lorsqu'il n'y a pas d'ambiguïté, on notera $N(x)$ pour $N_V(x)$. $N(x)$ est aussi appelé le voisinage ouvert de x.

NOTATION A.2 (Voisinage Fermé). Le voisinage fermé d'un sommet x, noté $N[x]$ correspond au voisinage ouvert de x plus l'élément $\{x\}$.

$$N[x] = N(x) \cup \{x\} \quad .$$

NOTATION A.3 (Voisinage et Graphes orientés). Dans un graphe orienté (ou un tournoi) $G = (V, A)$, nous noterons par $N^+(x)$ le voisinage sortant du sommet x et $N^-(x)$ le voisinage entrant de x.

NOTATION A.4 (Sous-graphe induit). Soit $G = (V, E)$ un graphe. Et soit X un sous-ensemble de V. Nous noterons $G[X]$ le sous-graphe de G induit par l'ensemble X : *i.e.* $H = (X, F) = G[X]$ où F est un sous-ensemble d'arête de E, où chacune des arêtes de F a ses deux extrémités dans X.

DÉFINITION A.5 (Fonction de Möbius). Fonction définie pour les entiers positifs :

$$\mu(n) = \begin{cases} 1 & \text{si } n = 1. \\ (-1)^k & \text{si } n \text{ est le produit de } k \text{ nombres premiers.} \\ 0 & \text{si } n \text{ contient un facteur carré.} \end{cases}$$

Table des figures

Bibliographie

A. V. Aho, J. E. Hopcroft et J. D. Ullman, *Analysis of Computer algorithms*, Assison-Wesley, 1974.

G. E. Andrews, *The theory of partitions*, Addison-Wesley, Isbn 0-201-13501-9, 1976.

S. Arnborg et A. Proskurowski, Linear time algorithms for np-hard problems restricted to partial k-trees , *Discrete Applied Mathematics*, vol. 23, n° 1, p. 11–24, 1989, doi : 10.1016/0166-218X(89)90031-0, adresse : http://dx.doi.org/10.1016/0166-218X(89)90031-0.

S. Arnborg, J. Lagergren et D. Seese, Easy problems for tree-decomposable graphs , *Journal of Algorithms*, vol. 12, n° 2, p. 308–340, 1991, doi : 10.1016/0196-6774(91)90006-K, adresse : http://dx.doi.org/10.1016/0196-6774(91)90006-K.

V. Arvind, B. Das et P. Mukhopadhyay, On isomorphism and canonization of tournaments and hypertournaments , Dans *ISAAC*, T. Asano (éditeur), vol. 4288 de *LNCS*, p. 449–459, Springer, 2006, Isbn 3-540-49694-7, doi : 10.1007/11940128_46, adresse : http://dx.doi.org/10.1007/11940128_46.

L. Babai et E. M. Luks, Canonical labeling of graphs , Dans *15th Annual ACM Symposium on Theory of Computing (STOC)*, p. 171–183, 1983, doi : 10.1145/800061.808746, adresse : http://doi.acm.org/10.1145/800061.808746.

J. Bang-Jensen et G. Gutin, *Digraphs : Theory, Algorithms and Applications*, Spinger, Isbn 1-85233-268-9, adresse : http://www.cs.rhul.ac.uk/books/dbook/, 2000.

A. Bernàth, A note on the directed source location algorithm , *submitter*, p. 9p, 2004, adresse : http://www.cs.elte.hu/egres/www/tr-04-12.html.

H. L. Bodlaender et U. Rotics, Computing the treewidth and the minimum fill-in with the modular decomposition , *Algorithmica*, vol. 36, n° 4, p. 375–408, 2003, doi : 10.1007/s00453-003-1026-5, adresse : http://dx.doi.org/10.1007/s00453-003-1026-5.

A. Bouchet, Un algorithme polynômial pour reconnaître les graphes d'alternance , *Comptes rendus de l'académie des sciences de Paris - Série A*, vol. 300, p. 569–572, 1985.

A. Bouchet, Reducing prime graphs and recognizing circle graphs , *Combinatorica*, vol. 7, n° 3, p. 243–254, 1987, doi : 10.1007/BF02579301, adresse : http://dx.doi.org/10.1007/BF02579301.

A. Brandstädt, V. B. Le et J. P. Spinrad, *Graph Classes : A Survey*, SIAM, ISBN 978-0-898714-32-6, 1999.

A. Bretscher, D. Corneil, M. Habib et C. Paul, A simple linear time lexbfs cograph recognition algorithm , *SIAM Journal on Discrete Mathematics*, vol. 22, n° 4, p. 1277–1294, 2008, doi : 10.1137/060664690, adresse : http://dx.doi.org/10.1137/060664690.

A. E. Brouwer,

The enumeration of locally transitive tournaments, 1980, Technical report ZW138 of stichting mathematisch centrum, Amsterdam.

B.-M. Bui-Xuan, *Tree-Representation of Set Families in Graph Decomposition and Efficient Algorithms*, Thèse de doctorat, Université Montpellier II, FRANCE, 2008.

B.-M. Bui-Xuan, M. Habib et C. Paul, Revisiting T. Uno and M. Yagiura's Algorithm , Dans *16th International Symposium of Algorithms and Computation (ISAAC05)*, D.-Z. D. Xiaotie Deng (éditeur), vol. 3827 de *Lecture Notes in Computer Science*, p. 146–155, Springer, 2005, doi : 10.1007/11602613, adresse : http://dx.doi.org/10.1007/11602613.

B.-M. Bui-Xuan, M. Habib, V. Limouzy et F. d. Montgolfier, On modular decomposition concepts : the case for homogeneous relations , *Electronic Notes in Discrete Mathematics*, vol. 27, p. 13–14, 2006a, doi : 10.1016/j.endm.2006.08.033, adresse : http://dx.doi.org/10.1016/j.endm.2006.08.033.

B.-M. Bui-Xuan, M. Habib, V. Limouzy et F. d. Montgolfier, Homogeneity vs. adjacency : generalising some graph decomposition algorithms , Dans *32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, F. V. Fomin (éditeur), vol. 4271 de *LNCS*, Springer, June 2006b, doi : 10.1007/11917496_25, adresse : http://dx.doi.org/10.1007/11917496_25.

B.-M. Bui-Xuan, M. Habib, V. Limouzy et F. d. Montgolfier, Unifying two graph decompositions with modular decomposition , Dans *Algorithms and Computation, 18th International Symposium, ISAAC 2007, Sendai, Japan, December 17-19, 2007*, T. Tokuyama (éditeur), vol. 4835 de *LNCS*, Springer, December 2007, doi : 10.1007/978-3-540-77120-3_7, adresse : http://dx.doi.org/10.1007/978-3-540-77120-3_7.

B.-M. Bui-Xuan, M. Habib, V. Limouzy et F. d. Montgolfier, Algorithmic aspects of a general modular decomposition theory , *Discrete Applied Mathematics*, vol. to appear, 2008a, adresse : http://hal.archives-ouvertes.fr/hal-00111235/fr/.

B.-M. Bui-Xuan, M. Habib, V. Limouzy et F. d. Montgolfier, A new tractable combinatorial decomposition. , *Submitted*, 2008b, adresse : http://hal-lirmm.ccsd.cnrs.fr/lirmm-00157502/fr/.

M. Burlet et J. Uhry, Parity graphs , *Annals of Discrete Mathematics*, vol. 16, p. 1–26, 1982.

C. Capelle, *Décomposition de Graphes et Permutations Factorisantes*, Thèse de doctorat, Université Montpellier II, 1997.

C. Capelle, M. Habib et F. d. Montgolfier, Graph decomposition and factorizing permutations , *Discrete Mathematics & Theoretical Computer Science*, vol. 5, n° 1, p. 55–70, 2002, adresse : http://www.dmtcs.org/dmtcs-ojs/index.php/dmtcs/article/view/161.

P. Charbit, M. Habib, V. Limouzy, F. d. Montgolfier, M. Raffinot et M. Rao, A note on computing set overlap classes , *Information Processing Letters*, 2008, doi : 10.1016/j.ipl.2008.05.005, adresse : http://dx.doi.org/10.1016/j.ipl.2008.05.005, to appear.

M. Chein, M. Habib et M.-C. Maurer, Partitive hypergraphs , *Discrete Mathematics*, vol. 37, n° 1, p. 35–50, 1981, doi : 10.1016/0012-365X(81)90138-2, adresse : http://dx.doi.org/10.1016/0012-365X(81)90138-2.

M. Chudnovsky, N. Robertson, P. D. Seymour et R. Thomas, The strong perfect graph theorem , *Annals of Mathematics*, vol. 164, n° 1, p. 51–229, 2006, adresse : http://annals.math.princeton.edu/issues/2006/July2006/ChudnovskyRobertsonSeymourThomas.pdf.

V. Chvátal, Star-cutsets and perfect graphs , *Journal of Combinatorial Theory, Series B*, vol. 39, n° 3, p. 189–199, 1985, doi : 10.1016/0095-8956(85)90049-8, adresse : http://dx.doi.org/10.1016/0095-8956(85)90049-8.

V. Chvátal et N. Sbihi, Bull-free berge graphs are perfect , *Graphs and Combinatorics*, vol. 3, n° 1, p. 127–139, 1987, doi : 10.1007/BF01788536, adresse : http://dx.doi.org/10.1007/BF01788536.

S. Cicerone et G. Di Stefano, On the extension of bipartite to parity graphs , *Discrete Applied Mathematics*, vol. 95, n° 1-3, p. 181–195, 1999a, doi : 10.1016/S0166-218X(99)00074-8, adresse : http://dx.doi.org/10.1016/S0166-218X(99)00074-8.

S. Cicerone et G. Di Stefano, Graph classes between parity and distance-hereditary graphs , *Discrete Applied Mathematics*, vol. 95, n° 1-3, p. 197–216, 1999b, doi : 10.1016/S0166-218X(99)00075-X, adresse : http://dx.doi.org/10.1016/S0166-218X(99)00075-X.

E. Clarou, *Une hiérarchie de forçage pour les tournois indécomposables*, Thèse de doctorat, Université Claude Bernard Lyon I, 1996.

C. J. Colbourn et D. G. Corneil, On deciding switching equivalence of graphs , *Discrete Applied Mathematics*, vol. 2, n° 3, p. 181–184, 1980, ISSN 0166-218X, doi : 10.1016/0166-218X(80)90038-4, adresse : http://dx.doi.org/10.1016/0166-218X(80)90038-4.

M. Conforti, G. Cornuéjols, A. Kapoor et K. Vuškovic, Even-hole-free graphs part I : Decomposition theorem , *Journal of Graph Theory*, vol. 39, n° 1, p. 6–49, 2001, doi : 10.1002/jgt.10006, adresse : http://dx.doi.org/10.1002/jgt.10006.

M. Conforti, G. Cornuéjols, A. Kapoor et K. Vuškovic, Even-hole-free graphs part II : Recognition algorithm , *Journal of Graph Theory*, vol. 40, n° 4, p. 238–266, 2002, doi : 10.1002/jgt.10045, adresse : http://dx.doi.org/10.1002/jgt.10045.

D. G. Corneil et R. Mathon, *Geometry and Combinatorics ; Selected Works of J.J. Seidel*, Academic Press, 1991.

D. G. Corneil, H. Lerch et L. Stewart Burlingham, Complement reducible graphs , *Discrete Applied Mathematics*, vol. 3, n° 3, p. 163–174, 1981, doi : 10.1016/0166-218X(81)90013-5, adresse : http://dx.doi.org/10.1016/0166-218X(81)90013-5.

D. G. Corneil, Y. Perl et L. Stewart, A linear recognition algorithm for cographs , *SIAM Journal on Computing*, vol. 14, n° 4, p. 926–934, 1985, doi : 10.1137/0214065, adresse : http://dx.doi.org/10.1137/0214065.

D. G. Corneil, M. Habib, J.-M. Lanlignel, B. A. Reed et U. Rotics, Polynomial time recognition of clique-width $\leqslant 3$ graphs (extended abstract) , Dans *LATIN*, G. H. Gonnet, D. Panario et A. Viola (éditeurs), vol. 1776 de *Lecture Notes in Computer Science*, p. 126–134, Springer, 2000, ISBN 3-540-67306-7.

G. Cornuéjols et W. H. Cunningham, Composition for perfect graphs , *Discrete Mathematics*, vol. 55, p. 245–254, 1985.

B. Courcelle, J. Engelfriet et G. Rozenberg, Handle-rewriting hypergraph grammars. , *Journal of Computer and System Sciences*, vol. 46, n° 2, p. 218–270, 1993, doi : 10.1016/0022-0000(93)90004-G, adresse : http://dx.doi.org/10.1016/0022-0000(93)90004-G.

A. Cournier et M. Habib, A new linear algorithm for modular decomposition , Dans *Trees in algebra and programming (CAAP 94)*, vol. 787 de *Lecture Notes in Computer Science*, p. 68–84, 1994.

W. H. Cunnigham et J. Edmonds, A combinatorial decomposition theory , *Canadian Journal of Mathematics*, vol. 32, p. 734–765, 1980.

W. H. Cunningham, *A combinatorial decomposition theory*, Thèse de doctorat, University of Waterloo, Waterloo, Ontario, Canada, 1973.

W. H. Cunningham, Decomposition of directed graphs , *SIAM Journal on Algebraic and Discrete Methods*, vol. 3, n° 2, p. 214–228, 1982, doi : 10.1137/0603021, adresse : http://dx.doi.org/10.1137/0603021.

E. Dahlhaus, Parallel algorithms for hierarchical clustering, and applications to split decomposition and parity graph recognition , *Journal of Algorithms*, vol. 36, n° 2, p. 205–240, 2000, doi : 10.1006/jagm.2000.1090, adresse : http://dx.doi.org/10.1006/jagm.2000.1090.

E. Dahlhaus, Efficient parallel modular decomposition (extended abstract) , Dans *WG*, M. Nagl (éditeur), vol. 1017 de *Lecture Notes in Computer Science*, p. 290–302, Springer,

1995, Isbn 3-540-60618-1, doi : 10.1007/3-540-60618-1_83, adresse : http://dx.doi.org/10.1007/3-540-60618-1_83.

E. Dahlhaus, J. Gustedt et R. M. McConnell, Efficient and practical modular decomposition , Dans *SODA*, p. 26–35, 1997.

E. Dahlhaus, J. Gustedt et R. M. McConnell, Efficient and practical algorithms for sequential modular decomposition , *Journal of Algorithms*, vol. 41, n° 2, p. 360–387, 2001, doi : 10.1006/jagm.2001.1185, adresse : http://dx.doi.org/10.1006/jagm.2001.1185.

E. Dahlhaus, J. Gustedt et R. M. McConnell, Partially complemented representations of digraphs. , *Discrete Mathematics & Theoretical Computer Science*, vol. 5, n° 1, p. 147–168, 2002.

R. Ducournau et M. Habib, La multiplicité de l'héritage dans les langages à objets , *Techniques et Science Informatique*, vol. 8, n° 1, p. 41–62, 1989.

J. Edmonds, Submodular functions, matroids, and certain polyhedra, Dans *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969)*, p. 69–87, Gordon and Breach, New York, 1970.

J. Edmonds et R. Giles, A min-max relation for submodular functions on graphs, Dans *Studies in integer programming (Proc. Workshop, Bonn, 1975)– Ann. of Discrete Math., Vol. 1*, p. 185–204, North-Holland, Amsterdam, 1977.

A. Ehrenfeucht et G. Rozenberg, Theory of 2-structures, part I : Clans, basic subclasses, and morphisms , *Theoretical Computer Science*, vol. 70, n° 3, p. 277–303, 1990a, doi : 10.1016/0304-3975(90)90129-6, adresse : http://dx.doi.org/10.1016/0304-3975(90)90129-6.

A. Ehrenfeucht et G. Rozenberg, Theory of 2-structures, part II : Representation through labeled tree families , *Theoretical Computer Science*, vol. 70, n° 3, p. 305–342, 1990b, doi : 10.1016/0304-3975(90)90130-A, adresse : http://dx.doi.org/10.1016/0304-3975(90)90130-A.

A. Ehrenfeucht et G. Rozenberg, Primitivity is hereditary for 2-structures , *Theoretical Computer Science*, vol. 70, n° 3, p. 343–358, 1990c, doi : 10.1016/0304-3975(90)90131-Z, adresse : http://dx.doi.org/10.1016/0304-3975(90)90131-Z.

A. Ehrenfeucht, H. N. Gabow, R. M. McConnell et S. 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*, vol. 16, n° 2, p. 283–294, 1994, doi : 10.1006/jagm.1994.1013, adresse : http://dx.doi.org/10.1006/jagm.1994.1013.

A. Ehrenfeucht, T. Harju et G. Rozenberg, *Theory of 2-Structures : A Framework for Decomposition and Transformation of Graphs*, World Scientific, Isbn 981-02-4042-2, adresse : http://www.wspc.com/books/mathematics/4197.html, 1999.

H. Everett, S. Klein et B. A. Reed, An algorithm for finding homogeneous pairs , *Discrete Applied Mathematics*, vol. 72, n° 3, p. 209–218, 1997, doi : 10.1016/0166-218X(95)00090-E, adresse : http://dx.doi.org/10.1016/0166-218X(95)00090-E.

M. G. Everett et S. Borgatti, Role colouring a graph , *Mathematical Social Sciences*, vol. 21, n° 2, p. 183–188, 1991, doi : 10.1016/0165-4896(91)90080-B, adresse : http://dx.doi.org/10.1016/0165-4896(91)90080-B.

J. Fiala et D. Paulusma, The computational complexity of the role assignment problem , Dans *Automata, Languages and Programming, 30th International Colloquium (ICALP)*, J. C. M. Baeten, J. K. Lenstra, J. Parrow et G. J. Woeginger (éditeurs), vol. 2719 de *Lecture Notes in Computer Science*, p. 817–828, Springer, June 30 - July 4 2003, doi : 10.1007/3-540-45061-0_64, adresse : http://dx.doi.org/10.1007/3-540-45061-0_64, Eindhoven, The Netherlands.

J. Fiala et D. Paulusma, A complete complexity classification of the role assignment problem , *Theoretical Computer Science*, vol. 349, n° 1, p. 67–81, 2005, doi : 0.1016/j.tcs.2005.09.029, adresse : http://dx.doi.org/10.1016/j.tcs.2005.09.029.

J.-L. Fouquet, V. Giakoumakis et J.-M. Vanherpe, Bipartite graphs totally decomposable by canonical decomposition , *International Journal of Foundations of Computer Science*, vol. 10, n° 4, p. 513–533, 1999, doi : 10.1142/S0129054199000368, adresse : http://dx.doi.org/10.1142/S0129054199000368.

S. Fujishige, *Submodular Functions and Optimization*, North-Holland, ISBN 9780444520869, 1991.

C. P. Gabor, W.-L. Hsu et K. J. Supowit, Recognizing circle graphs in polynomial time , *Journal of ACM*, vol. 36, n° 3, p. 435–473, 1989, doi : 10.1145/65950.65951, adresse : http://doi.acm.org/10.1145/65950.65951.

H. N. Gabow, A representation for crossing set families with applications to submodular flow problems , Dans *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 202–211, ACM/SIAM, 25–27 June 1993, doi : 313559.313753, adresse : http://doi.acm.org/313559.313753.

H. N. Gabow, Centroids, representations, and submodular flows , *Journal of Algorithms*, vol. 18, n° 3, p. 586–628, 1995, doi : 10.1006/jagm.1995.1022, adresse : http://dx.doi.org/10.1006/jagm.1995.1022.

J. Gagneur, R. Krause, T. Bouwmeester et G. Casari, Modular decomposition of protein-protein interaction networks , *Genome Biology*, vol. 5, n° 8, 2004, doi : 10.1186/gb-2004-5-8-r57, adresse : http://dx.doi.org/10.1186/gb-2004-5-8-r57.

T. Gallai, Transitiv orientierbare Graphen , *Acta Mathematica Academiae Scientiarum Hungaricae*, vol. 18, p. 25–66, 1967.

M. R. Garey et D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, Isbn 0716710455, 1979.

M. C. Golumbic, Algorithmic graph theory and perfect graphs, Dans *Annals of Discrete Mathematics*, vol. 57, p. 314, Elsevier, second édition, 2004, Isbn 0-444-51530-5.

M. Grötschel, L. Lovász et A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, Isbn 3-540-13624-X, 1988.

F. Gurski et E. Wanke, Minimizing nlc-width is np-complete. , Dans *WG*, D. Kratsch (éditeur), vol. 3787 de *Lecture Notes in Computer Science*, p. 69–80, Springer, 2005, Isbn 3-540-31000-2.

M. Habib et C. Paul, A simple linear time algorithm for cograph recognition , *Discrete Applied Mathematics*, vol. 145, n° 2, p. 183–197, 2005, doi : 10.1016/j.dam.2004.01.011, adresse : http://dx.doi.org/10.1016/j.dam.2004.01.011.

M. Habib, C. Paul et L. Viennot, Partition refinement techniques : An interesting algorithmic tool kit , *International Journal of Foundations of Computer Science*, vol. 10, n° 2, p. 147–170, 1999, doi : 10.1142/S0129054199000125, adresse : http://dx.doi.org/10.1142/S0129054199000125.

M. Habib, F. d. Montgolfier et C. Paul, A simple linear-time modular decomposition algorithm , Dans *9th Scandinavian Workshop on Algorithm Theory (SWAT04)*, vol. 3111 de *Lecture Notes in Computer Science*, p. 187–198, 2004, doi : 10.1007/b98413, adresse : http://dx.doi.org/10.1007/b98413.

R. B. Hayward, Recognizing P_3-structure : A switching approach , *Journal of Combinatorial Theory, Series B*, vol. 66, n° 2, p. 247–262, 1996, doi : 10.1006/jctb.1996.0018, adresse : http://dx.doi.org/10.1006/jctb.1996.0018.

A. Hertz, On perfect switching classes , *Discrete Applied Mathematics*, vol. 94, n° 1-3, p. 3–7, 1999, doi : 10.1016/S0166-218X(98)00153-X, adresse : http://dx.doi.org/10.1016/S0166-218X(98)00153-X.

J. E. Hopcroft et R. E. Tarjan, A V^2 algorithm for determining isomorphism of planar graphs , *Information Processing Letters*, vol. 1, n° 1, p. 32–34, 1971, doi : 10.1016/0020-0190(71)90019-6, adresse : http://dx.doi.org/10.1016/0020-0190(71)90019-6.

J. E. Hopcroft et R. E. Tarjan, A $V \log V$ algorithm for isomorphism of triconnected planar graphs , *Journal of Computer and System Sciences*, vol. 7, n° 3, p. 323–331, 1973, doi : 10.1016/S0022-0000(73)80013-3, adresse : http://dx.doi.org/10.1016/S0022-0000(73)80013-3.

J. E. Hopcroft et J. K. Wong, Linear time algorithm for isomorphism of planar graphs (preliminary report) , Dans *STOC*, p. 172–184, ACM, 1974, doi : 10.1145/800119.803896, adresse : http://doi.acm.org/10.1145/800119.803896.

W.-L. Hsu et T.-H. Ma, Substitution decomposition on chordal graphs and applications , Dans *2nd International Symposium on Algorithms (ISA91)*, vol. 557 de *LNCS*, p. 52–60, 1991, doi : 10.1007/3-540-54945-5_49, adresse : http://dx.doi.org/10.1007/3-540-54945-5_49.

W.-L. Hsu et R. M. McConnell, PC-trees and circular-ones arrangements , *Theoretical Computer Science*, vol. 296, n° 1, p. 99–116, 2003, doi : 10.1016/S0304-3975(02)00435-8, adresse : http://dx.doi.org/10.1016/S0304-3975(02)00435-8.

A. W. Ingleton, A note on independence functions and rank. , *Journal of the London Mathematical Society*, vol. s1-34, n° 1, p. 49–56, 1959, doi : 10.1112/jlms/s1-34.1.49, adresse : http://dx.doi.org/10.1112/jlms/s1-34.1.49.

S. Iwata, A faster scaling algorithm for minimizing submodular functions , Dans *IPCO*, W. Cook et A. S. Schulz (éditeurs), vol. 2337 de *Lecture Notes in Computer Science*, p. 1–8, Springer, 2002, ISBN 3-540-43676-6, doi : 10.1007/3-540-47867-1, adresse : http://dx.doi.org/10.1007/3-540-47867-1.

S. Iwata, A faster scaling algorithm for minimizing submodular functions , *SIAM Journal on Computing*, vol. 32, n° 4, p. 833–840, 2003, doi : 10.1137/S0097539701397813, adresse : http://dx.doi.org/10.1137/S0097539701397813.

S. Iwata, Submodular function minimization , *Mathematical Programming*, vol. 112, n° 1, p. 45–64, 2008, doi : 10.1007/s10107-006-0084-2, adresse : http://dx.doi.org/10.1007/s10107-006-0084-2.

S. Iwata, L. Fleischer et S. Fujishige, A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions , Dans *STOC*, p. 97–106, 2000, doi : 10.1145/335305.335317, adresse : http://doi.acm.org/10.1145/335305.335317.

S. Iwata, L. Fleischer et S. Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions , *Journal of ACM*, vol. 48, n° 4, p. 761–777, 2001, doi : 10.1145/502090.502096, adresse : http://doi.acm.org/10.1145/502090.502096.

Ö. Johansson, NLC_2-decomposition in polynomial time , *International Journal of Foundations of Computer Science*, vol. 11, n° 3, p. 373–395, 2000, doi : 10.1142/S0129054100000223, adresse : http://dx.doi.org/10.1142/S0129054100000223.

J. Kratochvíl, J. Nešetřil et O. Zýka, On the computational complexity of Seidel's switching, Dans *Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity (Prachatice, 1990)*, vol. 51 de *Annals of Discrete Mathematics*, p. 161–166, North-Holland, Amsterdam, 1992.

D. Kratsch, R. M. McConnell, K. Mehlhorn et J. P. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs , Dans *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 158–167, ACM/SIAM, 2003, doi : 10.1145/644108.644137, adresse : http://doi.acm.org/10.1145/644108.644137.

D. Kratsch, R. M. McConnell, K. Mehlhorn et J. P. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs , *SIAM Journal on Computing*, vol. 36, n° 2, p. 326–353, 2006, doi : 10.1137/S0097539703437855, adresse : http://dx.doi.org/10.1137/S0097539703437855.

J.-M. Lanlignel, *Autour de la Décomposition en Coupes*, Thèse de doctorat, Université Montpellier II, 2001.

G. Lopez et C. Rauzy, Reconstruction of binary relations from their restrictions of cardinality 2, 3, 4 and (n-1) I , *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 38, n° 1, p. 27–37, 1992a, doi : 10.1002/malq.19920380104, adresse : http://dx.doi.org/10.1002/malq.19920380104.

G. Lopez et C. Rauzy, Reconstruction of binary relations from their restrictions of cardinality 2, 3, 4 and (n-1) II , *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 38, n° 1, p. 157–168, 1992b, doi : 10.1002/malq.19920380111, adresse : http://dx.doi.org/10.1002/malq.19920380111.

T.-H. Ma et J. P. Spinrad, An $O(n^2)$ algorithm for undirected split decompositon , *Journal of Algorithms*, vol. 16, n° 1, p. 154–160, 1994.

N. V. R. Mahadev et U. N. Peled, Threshold graphs and related topics, Dans *Annals of Discrete Mathematics*, vol. 56, p. 543, Elsevier, 1995, Isbn 0-444-51530-5.

R. M. McConnell, A certifying algorithm for the consecutive-ones property , Dans *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 768–777, 2004, doi : 10.1145/982792.982909, adresse : http://doi.acm.org/10.1145/982792.982909.

R. M. McConnell, An $O(n^2)$ incremental algorithm for modular decomposition of graphs and 2-structures , *Algorithmica*, vol. 14, n° 3, p. 229–248, 1995, doi : 10.1007/BF01206330, adresse : http://dx.doi.org/10.1007/BF01206330.

R. M. McConnell et F. d. Montgolfier, Linear-time modular decomposition of directed graphs , *Discrete Applied Mathematics*, vol. 145, n° 2, p. 198–209, 2005a, doi : 10.1016/j.dam.2004.02.017, adresse : http://dx.doi.org/10.1016/j.dam.2004.02.017.

R. M. McConnell et F. d. Montgolfier, Algebraic Operations on PQ Trees and Modular Decomposition Trees , Dans *31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, vol. 3787, p. 421–432, 2005b, doi : 10.1007/11604686_37, adresse : http://dx.doi.org/10.1007/11604686_37.

R. M. McConnell et J. P. Spinrad, Ordered vertex partitioning , *Discrete Mathematics & Theoretical Computer Science*, vol. 4, n° 1, p. 45–60, 2000, adresse : http://www.dmtcs.org/dmtcs-ojs/index.php/dmtcs/article/view/113.

R. M. McConnell et J. P. Spinrad, Linear-time modular decomposition and efficient transitive orientation of comparability graphs , Dans *Proceedings of the Fifth Annual ACM-SIAM*

Symposium on Discrete Algorithms (SODA), p. 536–545, New York, ACM, 1994.

R. M. McConnell et J. P. Spinrad, Modular decomposition and transitive orientation , *Discrete Mathematics*, vol. 201, n° 1-3, p. 189–241, 1999, doi : 10.1016/S0012-365X(98)00319-7, adresse : http://dx.doi.org/10.1016/S0012-365X(98)00319-7.

R. H. Möhring, Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and boolean functions , *Annals of Operations Research*, vol. 6, p. 195–225, 1985.

R. H. Möhring et F. J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization , *Annals of Discrete Mathematics*, vol. 19, p. 257–356, 1984.

F. d. Montgolfier, *Décomposition modulaire des graphes. Théorie, extensions et algorithmes*, Thèse de doctorat, Université Montpellier II, 2003.

F. d. Montgolfier et M. Rao, The bi-join decomposition , *Electronic Notes in Discrete Mathematics*, vol. 22, p. 173–177, 2005a, doi : 10.1016/j.endm.2005.06.039, adresse : http://dx.doi.org/10.1016/j.endm.2005.06.039.

F. d. Montgolfier et M. Rao, Bipartitive families and the bi-join decomposition , *submitted*, 2005b, http://hal.archives-ouvertes.fr/hal-00132862.

S. Olariu, No antitwins in minimal imperfect graphs , *Journal of Combinatorial Theory, Series B*, vol. 45, n° 2, p. 255–257, 1988, doi : 10.1016/0095-8956(88)90071-8, adresse : http://dx.doi.org/10.1016/0095-8956(88)90071-8.

J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization , Dans *IPCO*, M. Fischetti et D. P. Williamson (éditeurs), vol. 4513 de *Lecture Notes in Computer Science*, p. 240–251, Springer, 2007, ISBN 978-3-540-72791-0, doi : 10.1007/978-3-540-72792-7_19, adresse : http://dx.doi.org/10.1007/978-3-540-72792-7_19.

S.-I. Oum, *Graphs Of Bounded Rank Width*, Thèse de doctorat, Princeton University, 2005.

J. Oxley, *Matroid Theory*, Oxford University Press, ISBN 0-19-920250-8, 1992, Reprinted in 2006.

R. Paige et R. E. Tarjan, Three partition refinement algorithms , *SIAM Journal on Computing*, vol. 16, n° 6, p. 973–989, 1987, ISSN 0097-5397, doi : 10.1137/0216062, adresse : http://dx.doi.org/10.1137/0216062.

C. Paul,

Aspects Algorithmiques de la Décomposition Modulaire,

Habilitation à Diriger des Recherches, Université Montpellier II, adresse : http://www.lirmm.fr/~paul/HdR/hdr.pdf, 2006.

C. Paul, *Parcours en largeur lexicographique : un algorithme de partitionnement, application aux graphes et généralisation*, Thèse de doctorat, Université Montpellier II, 1998.

F. Protti, M. D. da Silva et J. L. Szwarcfiter, Applying modular decomposition to parameterized bicluster editing , Dans *IWPEC*, H. L. Bodlaender et M. A. Langston (éditeurs), vol. 4169 de *Lecture Notes in Computer Science*, p. 1–12, Springer, 2006, Isbn 3-540-39098-7, doi : 10.1007/11847250_1, adresse : http://dx.doi.org/10.1007/11847250_1.

R. Rado, A theorem on independence relations , *Quarterly Journal of Mathematics*, vol. 13, n° 1, p. 83–89, 1942, doi : 10.1093/qmath/os-13.1.83, adresse : http://dx.doi.org/10.1093/qmath/os-13.1.83.

R. Rado, A note on independence functions , *Proceedings of the London Mathematical Society*, vol. s3-7, n° 1, p. 300–320, 1957, doi : 10.1112/plms/s3-7.1.300, adresse : http://dx.doi.org/10.1112/plms/s3-7.1.300.

M. Rao, Coloring a graph using split decomposition , Dans *WG*, J. Hromkovic, M. Nagl et B. Westfechtel (éditeurs), vol. 3353 de *Lecture Notes in Computer Science*, p. 129–141, Springer, 2004, Isbn 3-540-24132-9, doi : 10.1007/b104584, adresse : http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3353&spage=129.

M. Rao, *Décompositions de Graphes et Algorithmes Efficaces*, Thèse de doctorat, Université de Metz, 2006.

N. Robertson et P. D. Seymour, Graph minors. XX. wagner's conjecture. , *Journal of Combinatorial Theory, Series B*, vol. 92, n° 2, p. 325–357, 2004, doi : 10.1016/j.jctb.2004.08.001, adresse : http://dx.doi.org/10.1016/j.jctb.2004.08.001.

N. Robertson et P. D. Seymour, Graph minors. II. algorithmic aspects of tree-width , *Journal of Algorithms*, vol. 7, n° 3, p. 309–322, 1986a, doi : 10.1016/0196-6774(86)90023-4, adresse : http://dx.doi.org/10.1016/0196-6774(86)90023-4.

N. Robertson et P. D. Seymour, Graph minors. V. excluding a planar graph , *Journal of Combinatorial Theory, Series B*, vol. 41, n° 1, p. 92–114, 1986b, doi : http://dx.doi.org/10.1016/0095-8956(86)90030-4, adresse : http://dx.doi.org/10.1016/0095-8956(86)90030-4.

N. Robertson et P. D. Seymour, Graph minors. IV. tree-width and well-quasi-ordering , *Journal of Combinatorial Theory, Series B*, vol. 48, n° 2, p. 227–254, 1990, doi : 10.1016/0095-8956(90)90120-O, adresse : http://dx.doi.org/10.1016/0095-8956(90)90120-O.

N. Robertson et P. D. Seymour, Graph minors. X. obstructions to tree-decomposition , *Journal of Combinatorial Theory, Series B*, vol. 52, n° 2, p. 153–190, 1991, doi : 10.1016/0095-8956(91)90061-N, adresse : http://dx.doi.org/10.1016/0095-8956(91)90061-N.

I. Rusu et J. P. Spinrad, Forbidden subgraph decomposition , *Discrete Mathematics*, vol. 247, n° 1-3, p. 159–168, 2002, doi : 10.1016/S0012-365X(01)00173-X, adresse : http://dx.doi.org/10.1016/S0012-365X(01)00173-X.

A. Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time , *Journal of Combinatorial Theory, Series B*, vol. 80, n° 2, p. 346–355, 2000, doi : 10.1006/jctb.2000.1989, adresse : http://dx.doi.org/10.1006/jctb.2000.1989.

A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, vol. A-B-C, Springer, 2003.

J. J. Seidel, A survey of two-graphs, Dans *Colloquio Internazionale sulle Teorie Combinatorie (Rome, 1973), Tomo I*, p. 481–511. Atti dei Convegni Lincei, No. 17, Accad. Naz. Lincei, Rome, 1976, reprint in Corneil et Mathon (1991).

N. J. A. Sloane,

The on-line encyclopedia of integer sequences ,
http://www.research.att.com/~njas/sequences/.

E. Speckenmeyer, On feedback problems in digraphs , Dans *15th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, vol. 411 de *Lecture Notes in Computer Science*, p. 218–231, 14–16 June 1989.

M. Tedder, D. G. Corneil, M. Habib et C. Paul, Simpler linear-time modular decomposition via recursive factorizing permutations , Dans *ICALP (1)*, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir et I. Walukiewicz (éditeurs), vol. 5125 de *Lecture Notes in Computer Science*, p. 634–645, Springer, 2008, ISBN 978-3-540-70574-1, doi : 10.1007/978-3-540-70575-8_52, adresse : http://dx.doi.org/10.1007/978-3-540-70575-8_52.

K. Truemper, On Whitney's 2-isomorphism theorem for graphs , *Journal of Graph Theory*, vol. 4, n° 1, p. 43–49, 1980, doi : 10.1002/jgt.3190040106, adresse : http://dx.doi.org/10.1002/jgt.3190040106.

T. Uno et M. Yagiura, Fast algorithms to enumerate all common intervals of two permutations , *Algorithmica*, vol. 26, n° 2, p. 290–309, 2000.

J. Vygen, A note on schrijver's submodular function minimization algorithm , *Journal of Combinatorial Theory, Series B*, vol. 88, n° 2, p. 399–402, 2003, doi : 10.1016/S0095-8956(02)00047-3, adresse : http://dx.doi.org/10.1016/S0095-8956(02)00047-3.

E. Wanke, k-NLC graphs and polynomial algorithms , *Discrete Applied Mathematics*, vol. 54, n° 2-3, p. 251–266, 1994, doi : 10.1016/0166-218X(94)90026-4, adresse : http://dx.doi.org/10.1016/0166-218X(94)90026-4.

D. R. White et K. P. Reitz, Graph and semigroup homomorphisms on networks of relations , *Social Networks*, vol. 5, n° 2, p. 193–234, 1983, doi : 10.1016/0378-8733(83)90025-4,

adresse : http://dx.doi.org/10.1016/0378-8733(83)90025-4.

H. Whitney, 2-isomorphic graphs , *American Journal of Mathematics*, vol. 55, n° 1, p. 245–254, 1933, adresse : http://www.jstor.org/stable/2371127.

Résumé

La décomposition modulaire est une décomposition de graphes très étudiée depuis son introduction en 67 par Gallai. Cette décomposition s'est montrée être un outil très puissant tant d'un point de vue théorique que d'un point de vue algorithmique. Dans cette thèse nous abordons ces deux aspects. Nous proposons deux généralisations de la décomposition modulaire et nous systématisons l'usage d'une technique très connue en algorithmique, l'affinage de partitions, pour résoudre les problèmes soulevés par ces généralisations. Le mémoire se divise en deux parties, la première partie est consacrée à l'introduction et l'étude de deux généralisations de la décomposition modulaire. Pour ce faire, nous introduisons une nouvelle structure discrète : les *"relations homogènes"* qui abstrait la notion de voisinage pour ne retenir que la notion essentielle de *distinction*. Nous étudions les propriétés combinatoires de la décomposition modulaire sur cette structure et nous présentons des méthodes pour calculer cette décomposition efficacement. Ensuite, nous introduisons une nouvelle décomposition, la décomposition en *"umodules"*. Il s'agit d'adopter le point de vue dual de la décomposition modulaire.

La seconde partie présente des algorithmes efficaces pour résoudre les deux problèmes suivants. Le premier problème consiste à trouver les composantes de chevauchement d'une famille de partie d'un ensemble fini. Nous présentons un algorithme dû à Dahlhaus et simplifions ce dernier en remplaçant une procédure complexe par une technique d'affinage de partition. Le second algorithme reconnaît les graphes de largeur NLC-2 en temps $O(n^2m)$ et fournit une méthode de même complexité pour résoudre le problème de l'isomorphisme sur cette classe.

Mots clés

Décomposition modulaire, umodule, affinage de partition, composantes de chevauchement, algorithmes efficaces, largeur NLC.

Abstract

Modular decomposition is a graph decomposition extensively studied since its introduction in 67 by Gallai. This decomposition appears to be a powerful tool from both theoretical and algorithmic point of view. In this thesis we will deal with both of them. We present two generalizations of modular decomposition and we systematize the use of a well known algorithmic technique, partition refinement, to solve the problem raised by these generalizations. The thesis is divided into two parts. The first part is dedicated to the introduction and the study of the generalizations of modular decomposition. To do so, we introduce a new discrete structure, *"homogeneous relations"* which abstract the notion of neighborhood to conserve only the essential notion of *distinction*. We study combinatoric properties of modular decomposition on this structure and we provide efficient algorithms to compute this decomposition. We then introduce a new decomposition, the *"umodular"* decomposition. It is to consider the dual point of view from modular decomposition.

The second part presents algorithms to solve the following problems. The first problem is, given a family of subsets of a finite set, find efficiently the overlap components of this family. We present an algorithm due to Dahlhaus, and we simplify its approach, by replacing a complex and tedious algorithmic technique using partition refinement. Algorithms obtained are linear. The second problem is to recognize graphs of NLC width 2. We present a $O(n^2m)$ algorithm and we provide a technique to test in the same complexity isomorphism of NLC-2 graphs.

Keywords

Modular decomposition, umodule, partition refinement, overlap components, efficient algorithms, NLC width